UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE FÍSICA

Teste e Caracterização do Subsistema de Controlo da Carta de Distribuição de Alta Tensão do Calorímetro TileCal/ATLAS

Cátia Sofia Pinto Silva Rato

Mestrado Integrado em Engenharia Física

Dissertação orientada por:
Professora Doutora Guiomar Evans

2017

Dedicatória e Agradecimentos

Ser um Engenheiro Físico implica entender que somos versáteis. Considerando todas as profissões científicas que existem, penso que nós somos aqueles com mais margem de manobra. Durante cinco anos somos iniciados em todos os assuntos relevantes à Física. Aprendemos matemática pura e usamo-la para provar tudo. Aprendemos programação e usamo-la para simular sistemas que nos permitem aprender mais. Aprendemos mecânica, relatividade, física nuclear, de partículas e atómica. Aprendemos a manobrar instrumentos que nos permitem ver mundos invisíveis ao olho humano. Após absorvermos todo o conhecimento que nos é possível, é-nos dada a escolha sobre o que escrever a nossa tese de mestrado. Escolhermos um tema não tão próximo do núcleo científico da Física como o esperado, não significa que não foi retido nada do que aprendemos. Significa que escolhemos dar uso à versatilidade que nos ensinam durante os cinco anos de estudo. Um Engenheiro Físico tem a capacidade de aprender sobre tudo. Esta dissertação de mestrado permite a culminação de eletrónica e programação, a liderarem um percurso gerido pela Física, que não foi nem será, de todo, esquecido.

Primeiramente, quero agradecer à minha família pela compreensão e apoio demonstrados ao longo dos últimos cinco anos. Aos meus colegas de curso, porque um curso universitário é muito melhor sucedido mediante a cooperação entre quem o frequenta. À professora Guiomar Evans por me ter dado a oportunidade de entrar num mundo único, quando eu ainda estava no 4º ano de faculdade, e por me ter apoiado em todos os passos do caminho desde então. Ao professor José Augusto por estar sempre disponível para as minhas dúvidas. Ao professor Agostinho Gomes pelo apoio fornecido ao longo do projeto e durante o estágio de três semanas que tive a oportunidade de frequentar no CERN. Ao Luís Gurriana e ao Filipe Martins. Ao LIP pela oportunidade de participação num projeto de tamanha importância como este. À Fundação da Faculdade de Ciências da Universidade de Lisboa pela bolsa de investigação que me foi atribuída no âmbito do projeto. A todos os funcionários da Universidade de Lisboa, quer sejam professores, funcionários da secretaria, biblioteca, bar ou seguranças, por todo o apoio fornecido ao longo dos últimos cinco anos e por todos os conhecimentos que me foram transmitidos. A todos os meus colegas de trabalho, na DXC Technology e na ES Field Delivery, por compreenderem a minha posição enquanto estudante de mestrado e por todo o conhecimento que me transmitem todos os dias.

Por último, o maior agradecimento de todos é dirigido à minha sobrinha, Carolina Rato, que continuamente me lembra da razão pela qual eu entrei neste curso e que todos os dias me ensina algo novo.

Um enorme obrigado a todos vocês.

Resumo

O trabalho apresentado nesta dissertação insere-se na colaboração portuguesa no projeto ATLAS/CERN, que consiste no desenvolvimento de um sistema de alta tensão remoto para substituir o atual sistema de alta tensão do calorímetro hadrónico TileCal, o calorímetro central da experiência ATLAS.

O sistema de alta tensão tem como objetivo a distribuição da alta tensão aos fotomultiplicadores do TileCal. Atualmente, este sistema está situado na caverna do ATLAS, onde está submetido a elevados níveis de radiação que gradualmente afetam o funcionamento dos seus componentes. Além disso, os níveis de radiação na caverna dificultam a reparação ou substituição dos componentes. A reformulação do sistema de alta tensão passa pela sua retirada da caverna do ATLAS, passando o sistema a ser remoto. Uma revisão completa da eletrónica utilizada é necessária, para atualizar os componentes obsoletos bem como para acomodar as melhorias a realizar no LHC. A carta responsável pela distribuição das altas tensões bem como das operações de leitura, controlo e calibração é designada por HV Opto. Com as alterações a efetuar a este sistema, esta carta passará a denominar-se HV Remote.

Em conjunto com a reformulação da eletrónica, é necessário proceder ao desenvolvimento de uma interface de utilizador que permita o seu controlo. Através desta interface, um utilizador será capaz de testar os componentes da HV Remote, bem como controlar o envio de dados para a carta. Para permitir a realização de testes aos componentes utilizados na placa HV Remote, foi desenhada e implementada uma placa de controlo dedicada, com os mesmos componentes da HV Remote, mas em menor quantidade, e sem as altas tensões. Além de servir para efetuar os testes de desempenhos aos vários componentes escolhidos para este sistema, a placa de controlo também permite efetuar o teste da interface de utilizador.

O objetivo desta dissertação consistiu no desenvolvimento da placa de controlo e da respetiva interface de utilizado e nos respetivos testes.

Palavras-chave: CERN, ATLAS, TileCal, Sistema de Controlo, Teste de Conversores, Desenho de PCBs (sigla de *Printed Circuit Boards*).

Abstract

The work presented in this dissertation is inserted on the Portuguese cooperation on the ATLAS/CERN project. It consists of the development of a remote high voltage system to replace the current high voltage system of the hadronic calorimeter TileCal, the central calorimeter of the ATLAS experiment.

The high voltage system's objective is the distribution of high voltage to the photomultipliers of TileCal. Nowadays, the system is on the ATLAS cave, where it is submitted to high amounts of radiation which gradually affect its components operation. Besides that, the high radiation levels hamper any intervention, like a repair or substitution of components that may arise. The reformulation of the system implies that the system will be moved out the ATLAS cave, therefore the system becomes remote. A complete revision of the actual electronics is needed, to upgrade the obsolete components and to accommodate the predicted upgrades of the LHC as well. The board responsible for the distribution of the high voltages and operations like reading, control or calibration is named HV Opto. With the predicted alterations to the system, the board will be named HV Remote.

Together with the reformulation of the electronics, it is necessary to develop a user interface that allows the control of said electronics. Through this interface, a user will be able to test every HV Remote components, being also able to control the high voltage values sent and read from the photomultipliers. To test the HV Remote components, it was designed and implemented a dedicated control board, with the same components of HV Remote, but in less quantity, and without the high voltages. Besides being used to test de component performance, the control board also allows the test of the user interface.

The work of this dissertation consisted of the development of two control boards, the user interface used to control de boards and the test of analog-to-digital converters present on the board.

Keywords: CERN, ATLAS, TileCal, Control System, Converters Testing, PCBs (*Printed Circuit Boards)* Design.

ÍNDICE

Índice de Figuras

Índice de Tabelas

Lista de Siglas

ADCs – *Analog to Digital Converters*

ALICE – *A Large Ion Collider*

ATLAS – *A Toroidal LHC ApparatuS*

CAN – *Controller Area Network*

CDF – *Cumulative Distribution Function*

CMOS – *Complementary Metal-Oxide Semiconductor*

CMS – *Compact Muon Solenoid*

CPHA – *Clock Phase*

CPOL – *Clock Polarity*

CS – *Chip Select*

DACs – *Digital to Analog Converters*

DNL – *Differential Non-Linearity*

EEPROM – *Electrically Erasable Programmable Read-Only Memory*

ENOB – *Effective Number Of Bits*

FPGA – *Field-Programmable Gate Array*

HL-LHC – *High Luminosity Large Hadron Collider*

HV – *High Voltage*

HVDS – *High Voltage Distributor System*

INL – *Integral Non-Linearity*

LHC – *Large Hadron Collider*

LHCb – *Large Hadron Collider beauty*

LSB – *Least Significant Bit*

LSBFIRST – *Least Significant Bit First*

MISO – *Master In Slave Out*

MOS – *Metal-Oxide Semiconductors*

MOSI – *Master Out Slave In*

MSBFIRST – *Most Significant Bit First*

PCB – *Printed Circuit Board*

PDF – *Probability Density Function*

PDIP – *Package Dual In-Line Pin*

PMTs – *Photomultipliers*

PWM – *Pulse Width Modulation*

RMS – *Root Mean Square*

SCLK – *Signal Clock*

SNR – *Signal-To-Noise Ratio*

SPI – *Serial Peripheral Interface*

SS – *Slave Select*

TileCal – *Tile Calorimeter*

UART – *Universal Asynchronous Receiver Transmitter*

VME – *Versa Module Eurocards*

1 INTRODUÇÃO

O LHC é o maior e mais potente acelerador de partículas do mundo, sendo as partículas aceleradas ao longo de um túnel circular, com 27 km, que se encontra a uma profundidade entre 50 e 175 m da superfície terrestre [1]. No interior deste acelerador, dois feixes de alta energia propagam-se em dois tubos separados (que possuem um nível de vácuo bastante elevado, entre 10^{-10} e 10^{-11} mbar [2]), em direções opostas, e com valores de velocidade perto dos da luz, sendo guiados ao longo do percurso do acelerador por um forte campo magnético mantido por ímanes supercondutores. O acelerador é controlado através do *CERN Control Centre*, sendo provocadas as colisões dos feixes em quatro posições do anel acelerador que correspondem às localizações de quatro detetores de partículas: ATLAS, CMS, ALICE e LHCb [3].

O TileCal é o calorímetro hadrónico central da experiência ATLAS que se encontra na caverna do ATLAS, aproximadamente a 100 metros de profundidade, sendo constituído por placas de aço intercaladas, periodicamente, com telhas plásticas cintiladoras que cobrem a maior parte da região central do detetor. É uma unidade independente, com toda a eletrónica associada contida no próprio calorímetro. Este é construído em três secções: um cilindro central dividido em duas partições, no interior do qual ocorrem as colisões, e dois cilindros expandidos. Cada uma das secções é dividida, segundo o ângulo azimutal, em 64 módulos [4].

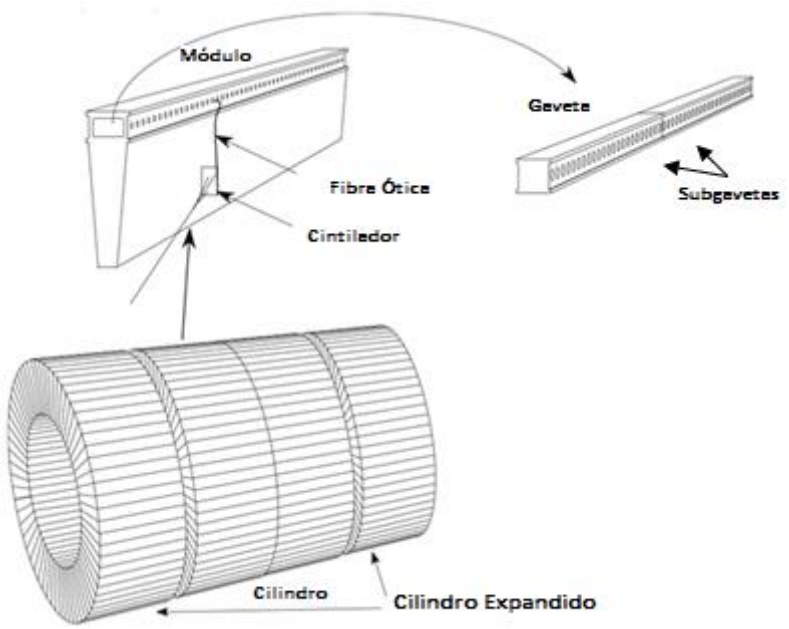


Figura 1.1 - Estrutura do calorímetro hadrónico TileCal.

A interação de partículas de alta energia com o aço produz jatos de partículas de energia mais baixa que provocam a emissão de fotões nos cintiladores. Esta radiação luminosa é transmitida através de fibras óticas, distribuídas radialmente e ligadas às arestas dos cintiladores, para os tubos fotomultiplicadores (PMTs[1]) [4]. Telhas adjacentes, em conjunto com as fibras óticas, são agrupadas para formar as células do TileCal. A informação que se propaga nas fibras óticas de cada célula é recolhida na extremidade das fibras, por dois PMTs. Isto significa que existem duas trajetórias que originam sinais elétricos diferentes,

[1] Sigla de *photomultipliers*.

permitindo a existência de redundância na recolha da informação e o aumento da sua uniformidade espacial [5]. O sinal analógico proveniente de cada PMT é processado por eletrónica que se situa na extremidade do detetor (numa gaveta por módulo, ver Figura 1.1), designada por Eletrónica de Frontaria e que é responsável pelo condicionamento e amplificação do sinal, fornecendo três sinais analógicos de saída: dois para o detetor de leitura e outro para o sistema de seleção/armazenamento de dados (*trigger*) [6].

Das experiências efetuadas no LHC, com colisões protão-protão de energias até 8 TeV, foram obtidos resultados relevantes como a descoberta do Bosão de Higgs e medições de processos do modelo Standard. Nos próximos anos pretende-se que a energia e a luminosidade do acelerador sejam aumentadas por fases, com o objetivo de criar o LHC de alta luminosidade (conhecido como HL-LHC) [7]. Prevê-se que os objetivos finais pretendidos sejam atingidos entre 2024 e 2035. Este projeto de atualização e melhoramento representa alguns desafios para os detetores do acelerador e para a respetiva eletrónica pois as novas condições operacionais implicam um nível de radiação superior, o que pode provocar uma falha parcial ou mesmo completa desses dispositivos.

O TileCal é um dos detetores afetados pelo aumento do nível de radiação. Este fator associado à obsolescência de alguns dos componentes utilizados na eletrónica dedicada bem como a algumas falhas verificas durante as experiências efetuadas implicam uma revisão de toda a eletrónica utilizada no TileCal. Especificamente, a colaboração portuguesa para o projeto ATLAS-CERN está encarregue da implementação de um sistema remoto de distribuição das altas tensões aos PMTs do calorímetro. O desenvolvimento deste sistema engloba três tarefas principais: a implementação do próprio sistema de alta tensão, a implementação de uma interface de utilizador para comunicar com o sistema e a implementação de uma carta de controlo, sem altas tensões, para verificar qual a sua influência no sistema principal. O trabalho desta dissertação de mestrado incide sobre as últimas duas tarefas, sendo a interface de utilizador descrita no Capítulo 3 e a carta de controlo no Capítulo 2.

1.1 SISTEMA DE DISTRIBUIÇÃO DE ALTA TENSÃO ATUAL

A principal função do sistema de distribuição de alta tensão *(HVDS*[2]) é, tal como o nome indica, distribuir a alta tensão a todos os PMTs do TileCal. Atualmente, este sistema encontra-se na caverna do ATLAS, junto ao detetor, e, consequentemente, está sujeito a doses elevadas de radiação que afetam gradualmente o funcionamento dos seus componentes. Os atuais elevados níveis de radiação na caverna dificultam a reparação ou substituição de eletrónica.

Toda a eletrónica de frontaria e de digitalização encontra-se compactada em gavetas removíveis que podem conter até 48 PMTs [8]. Cada gaveta é constituída por suas subgavetas (ver Figura 1.1), cada uma das quais pode conter até 24 blocos PMTs e eletrónica dedicada a cada bloco. No âmbito dessa dissertação considera-se apenas a carta responsável pela distribuição das altas tensões dedicadas a cada PMT e respetiva eletrónica de leitura, controlo e calibração, designada por HV Opto. A informação digital envolvida nos processos de leitura e controlo é transferida para unidades de controlo adicionais que se encontram na sala de controlo do calorímetro, fora da caverna do ATLAS, sendo a transferência de dados efetuada com recurso a fibras óticas.

As altas tensões individuais dos PMTs são definidas tendo por base um de dois valores de alta tensão (-830 V ou -950 V) fornecidos por uma fonte externa. Estes dois valores distintos de alta tensão permitem compensar as perdas de sinal devidas à degradação dos componentes óticos (cintiladores e fibras),

[2] Sigla de *High Voltage Distributor System*

causada pela radiação [5]. As duas tensões podem ser ajustadas localmente através da carta HV Opto, que permite ajustar até 24 tensões diferentes, numa faixa de 350 V abaixo da tensão de entrada aplicada [9]. Para cada duas HV Opto existe uma carta de controlo dedicada, a HV Micro, desenhada em torno de um microcontrolador (Motorola MC68376). A comunicação entre a HV Opto e a HV Micro é efetuada por um barramento VME[3]. As fontes de alta tensão estão localizadas na sala de controlo, fora da caverna do ATLAS, sendo utilizado o barramento CAN[4] na comunicação entre o sistema de controlo e as cartas HV Opto do TileCal.

Cada gaveta do TileCal dispõe de duas HV Opto que permitem as seguintes funcionalidades:

1. Ligar/Desligar os canais pares ou ímpares para cada conjunto de 24 PMTs;
2. Ajuste individual, para cada PMT, do valor de alta tensão de entrada – seis conversores digital-analógico (DACs)[5], de quatro canais, fornecem a tensão de entrada à malha de regulação. Cada uma das 24 malhas de regulação é realizada com recurso a um acoplador ótico de alta tensão;
3. Leitura individual do valor de alta tensão aplicado ao PMT – bloco constituído por dois multiplexadores analógicos de 16 entradas, um amplificador de instrumentação e um conversor analógico-digital (ADC)[6];
4. Leitura de dois sensores de temperatura;
5. Leitura de uma tensão de referência para teste;
6. Armazenamento dos parâmetros de inicialização da HV Opto numa *EEPROM*[7] de 2 kB (AT25160).

Existem também dois barramentos de alta tensão (HV[8]) que servem de suporte às duas HV Opto e à HV Micro, distribuindo também as tensões ajustadas aos respetivos PMTs. Por último, existe um barramento flexível, que permite a ligação entre os barramentos das duas subgavetas [10]. Este esquema está representado na Figura 1.2.

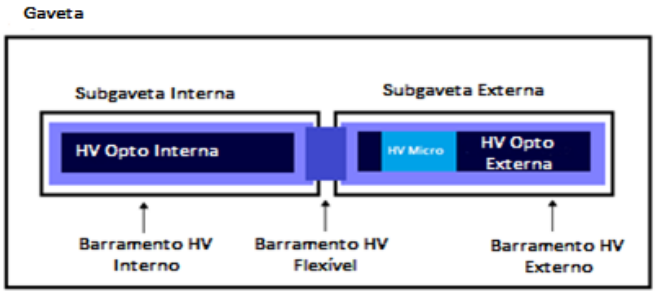


Figura 1.2 - Esquema mecânico do sistema de alta tensão no interior de uma gaveta.

Os parâmetros de configuração de cada HV Opto, armazenados na EEPROM, são lidos pelo microcontrolador (32-*bit* MC68376), contido na HV Micro, e guardados numa memória RAM de 256 kB (HITACHI HM62128) disponível nessa carta. A tensão a aplicar a cada PMT é enviada aos conversores digital-analógico (DACs), que fornecem o sinal analógico de entrada das malhas de

[3] Sigla de *Versa Module Eurocards*, barramento paralelo e assíncrono que permite a coexistência de vários microcontroladores. É constituído por 11 linhas com transferência de dados de 8 a 32 *bits* a uma taxa máxima de 30 a 40 Mb/s.
[4] Sigla de *Controller Area Network*, é um protocolo padrão de comunicação série síncrona de elevada imunidade ao ruído e que utiliza apenas duas linhas para a comunicação. É utilizado para a comunicação entre microcontroladores e dispositivos sem a intervenção de um computador.
[5] Sigla de *Digital-to-Analog Converters*.
[6] Sigla de *Analog-to-Digital Converter*.
[7] Sigla de *Electrically Erasable Programmable Read-Only Memory*.
[8] Sigla de *High Voltage*.

regulação. Esta tensão analógica é amplificada e permite ajustar a tensão do respetivo PMT para o valor pretendido. O processo de leitura da HV atual e de cada PMT é efetuado com recurso a um divisor de tensão, por um fator de 1/1000, e a um conversor analógico-digital (ADC). O processo de leitura utiliza o mesmo barramento VME do controlo da carta, sendo os dados processados no microcontrolador disponível na HV Micro.

1.2 ATUALIZAÇÃO E MELHORIA DO TILECAL

Como foi referido anteriormente, os planos para aumentar a luminosidade do LHC implicam o aumento dos níveis de radiação, o que por sua vez implica a redução da acessibilidade ao calorímetro bem como a alteração da eletrónica associada para comportar os elevados níveis de radiação. Por outro lado, o projeto e a implementação da eletrónica foram realizados no final dos anos 90. Apesar de melhorias terem sido efetuadas ao longo dos anos, deve considerar-se ainda o envelhecimento e até o não funcionamento dos componentes devido à radiação a que foram submetidos, bem como a possibilidade de se terem tornado obsoletos, impossibilitando a sua utilização no projeto de atualização e melhoria do TileCal. Todos estes fatores, juntamente com a intenção de melhorar a eficiência do sistema a nível geral (nomeadamente, através da mitigação do ruído, aumentar a rapidez de aquisição e armazenamento bem como a resistência à radiação), implicam que a eletrónica seja toda verificada e redesenhada. Os PMTs serão, em princípio, os únicos componentes a serem mantidos.

Estão a ser desenvolvidas duas alternativas possíveis: uma análoga à atual, mas com atualização dos componentes eletrónicos, e outra com um sistema de HV remoto. Em ambas as abordagens, os divisores passivos dos PMTs são substituídos por divisores ativos. Outra alteração a ser realizada é a opção de todas as HV dos PMTs serem desligadas de forma individual em vez serem desligadas em grupos de 12 PMTs, como se verifica atualmente.

A equipa portuguesa, na qual se insere esta dissertação, está encarregue da abordagem remota, procedendo-se de seguida a uma descrição sucinta desse sistema.

1.2.1 Sistema de Alta Tensão Remoto

Contrariamente ao atual sistema, situado na caverna do ATLAS e sujeito a níveis elevados de radiação, o sistema de distribuição de alta tensão remoto que está a ser projetado estará localizado na caverna de eletrónica, designada por USA15, localizada a 100 m da caverna do ATLAS. Então, dois dos fatores de grande preocupação com o sistema eletrónico atual deixam de ser importantes: o sistema deixará de estar submetido a altos níveis de radiação e o seu acesso passará a ser permanente [11]. Este sistema tem, no entanto, a desvantagem de necessitar de um elevado número de cabos multi-condutores. Para além do já referido, a realização deste sistema remoto terá em conta o seguinte:

1. A carta HV Opto passará a ser denominada por HV Remote;
2. As HV de entrada para cada subgaveta, onde estará inserida uma HV Remote, serão distribuídas por cabos multi-condutores, de cerca de 100 m, com um cabo por subgaveta;
3. A distribuição de alta tensão para os PMTs será realizada através de um barramento de HV passivo equipado com supressores de ruído;
4. A revisão da malha de regulação para mitigar o ruído;

5. A comunicação entre todas as HV Remote e o sistema de controlo passará a ser feita através do protocolo SPI[9]-Ethernet, sendo o sistema de controlo do HVDS, implementado em Python e Unix, posteriormente incorporado no sistema de controlo do calorímetro TileCal.

Um dos pontos-chave do novo sistema é a malha de regulação, cujo elemento principal é um acoplador de alta tensão (Motorola MOC8204). Como os PMTs vão utilizar divisores ativos, a associação entre eles e os cabos longos provocam um aumento do ruído na alta tensão, efeito que será amplificado se as malhas de regulação tiverem dois transístores, como no esquema atual (Figura 1.3). Através da realização de vários testes, foi escolhido um novo esquema para a malha de regulação, sem transístores, que também pode ser observado na Figura 1.3 e que permite a mitigação do ruído. O outro ponto-chave é a carta HV Micro deixar de ser necessária, sendo o controlo do HVDS efetuado com recurso a um computador. A eletrónica de frontaria também deixará de estar sujeita aos elevados níveis de radiação previstos com o aumento da luminosidade do HL-LHC.

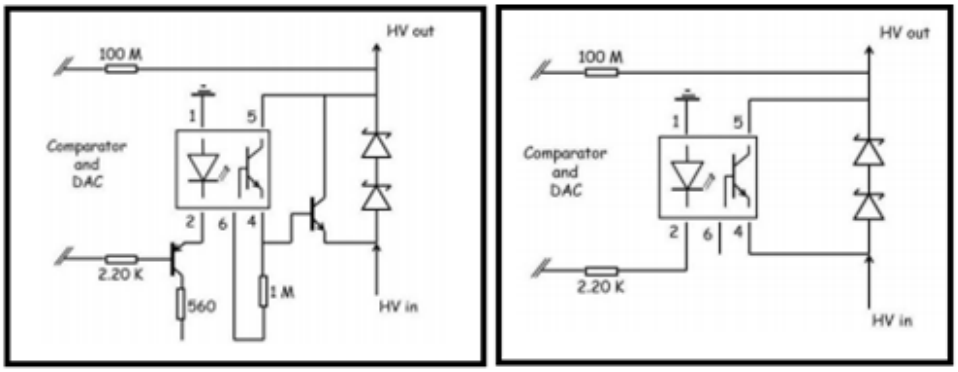


Figura 1.3 - Malha de regulação atual (à esquerda) e a proposta (à direita).

Uma das fases de trabalho desta dissertação consistiu na familiarização com o HVDS atual e com as alterações provocadas pelos requisitos para a sua atualização, tendo sido, posteriormente, elaborada uma placa de controlo, sem alta tensão, da HV Remote. A HV Remote foi projetada e está a ser implementada simultaneamente. A carta de controlo da HV Remote será utilizada para testes de desempenho do sistema de controlo e de leitura da HV Remote e, simultaneamente, para teste de linearidade dos DACs e ADCs utilizados, sem interferência dos sinais de alta tensão.

Após receção da carta de controlo, da sua montagem e testes iniciais foram verificados alguns erros na sua conceção e a necessidade de adição de pontos de teste em todos os sinais. Acordou-se a realização de um novo protótipo que inclui algumas alterações face à primeira versão da carta de controlo. Devido à janela temporal de fabrico e assemblagem do segundo protótipo comparativamente a esta dissertação foi ainda decidido que se procederia ao teste da interface de controlo da placa através do teste de componentes individuais, semelhantes aos utilizados na placa de controlo.

Nos próximos capítulos será descrita a placa de controlo da HV Remote (Capítulo 2) e o seu funcionamento, seguindo-se a descrição de geradores de ruído digitais para o teste automático de ADCs e DACs (Capítulo 3) e por último efetua-se a descrição da interface de utilizador que controla a placa de controlo da HV Remote (Capítulo 4) bem como os testes efetuados e respetivos resultados (Capítulo

[9] Sigla de *Serial Peripheral Interface*, protocolo de comunicação série síncrona constituído por duas linhas de dados (*full-duplex*) e duas linhas de controlo, uma dedica ao sinal de relógio e outra ao endereçamento.

5). Esta dissertação termina com as conclusões retiradas do trabalho efetuado e trabalho a realizar futuramente (Capítulo 6).

2 PLACA DE CONTROLO DA HV REMOTE

Uma das maiores preocupações no projeto da HV Remote é o ruído, como tal, todos os componentes devem ser testados individualmente de forma a conhecer a sua influência neste parâmetro. Considerando que o ruído pode provir da própria alta tensão associada à HV Remote, foi projetada uma placa, com as mesmas funcionalidades da HV Remote mas que apenas envolve os sinais de controlo digitais e leitura dos sinais analógicos. A placa, designada por HV Opto's Control[10], é composta por uma parte digital e outra analógica, com os mesmos componentes e funcionalidades da HV Remote mas com um número reduzido de componentes.

Como referido anteriormente, após testes iniciais à placa de controlo verificaram-se alguns erros no desenho da mesma e mau acesso a alguns sinais cuja observação é necessária no decorrer dos testes. Como tal, foi elaborado um segundo protótipo que também será descrito neste capítulo. A diferença entre o primeiro e o segundo protótipo é a introdução de pontos de teste em todos os sinais, bem como a inclusão de um novo DAC da Maxim, MAX5725, para testar qual o DAC mais adequado ao funcionamento da placa HV Remote. Esta inclusão deveu-se ao facto de ter sido no DAC que se observou um comportamento não esperado ao efetuar o teste do 1º protótipo. O DAC do primeiro protótipo, DAC7568 da Texas Instruments, foi mantido pois o comportamento observado poderia dever-se a anomalias no próprio DAC ou a problemas não detetados durante a assemblagem do protótipo.

O esquema elétrico, bem como os seus PCBs[11], foram realizados no âmbito desta dissertação com recurso ao *software Altium Designer 13.2* e, juntamente com as listas de material, encontram-se nos anexos 8.1 ao 8.8.

2.1 FUNCIONAMENTO DA PLACA DE CONTROLO

O diagrama de blocos da placa de controlo do 1º protótipo encontra-se apresentado na Figura 2.1.

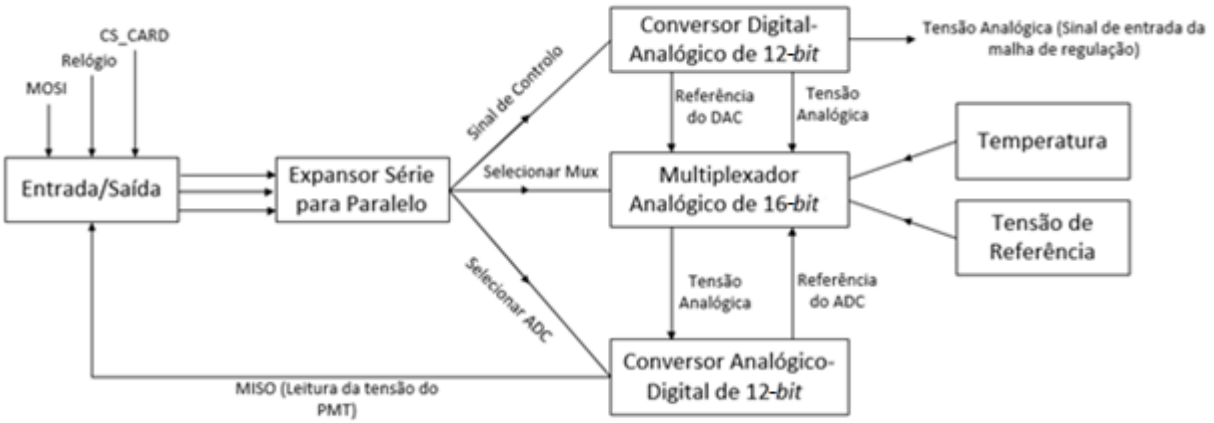


Figura 2.1 - Esquema funcional da placa de controlo da HV Remote.

[10] O nome da placa de controlo foi decidido antes da alteração do nome da carta de alta tensão de HV Opto para HV Remote, estando impresso na própria carta.
[11] Sigla de *Printed Circuit Board.*

A comunicação entre a placa de controlo e o computador utiliza dois protocolos de comunicação e encontra-se esquematizada na Figura 2.2.

Figura 2.2 - Comunicação entre um computador e a placa de controlo.

Inicialmente, a comunicação entre o computador e a placa de controlo seria feita com recurso a um módulo comercial desenvolvido pela Tibbo, EM1206+RJ203, que permitiria o envio de dados por Ethernet para o módulo e o seu posterior envio por SPI para a placa de controlo. Devido a problemas logísticos na encomenda do módulo, optou-se por utilizar um Arduino Uno e recorrer ao protocolo série entre o computador e o Arduino para efetuar a primeira parte do processo de comunicação, mantendo-se o protocolo SPI na segunda parte do processo. A escolha do protocolo série SPI teve por base o seu reduzido número de linhas, sendo os dados de controlo (saída do DAC) e a leitura das tensões dos PMTs (saída do ADC) enviados em 2 *bytes* consecutivos através das respetivas linhas de dados (MOSI e MISO[12]).

Esta placa de controlo contém um conversor DC/DC MAX3002 (designado por M1), um expansor série (SPI)/paralelo de 16-*bit* MCP23S17 (designado por M2), um DAC de 12-*bit* e 8 canais DAC7568 (designado por M3), um multiplexador analógico de 16-*bit* MPC506 (designado por M4), um amplificador de instrumentação INA128 (designado por M5), um ADC de 12-*bit* TLV2541 (designado por M6), uma referência de tensão REF02 (designada M7), um sensor de temperatura TMP17 (designado por M8) e uma tensão de referência AD589 (designada por DZ1). O esquema elétrico da placa de controlo encontra-se nos Anexos 8.1 e 8.2.

A placa de controlo foi projetada tendo em vista a utilização do módulo da Tibbo, cujos sinais digitais SPI são alimentados a 3.3 V, contudo, na HV Remote os sinais digitais são sinais CMOS[13] de 5.0 V. Como tal, foi integrado no circuito da placa um conversor DC/DC (M1), ver Figura 2.3, que permitiria a conversão desses sinais para sinais CMOS de 5.0 V. Com a alteração mencionada anteriormente, este componente deixa de ser fundamental na placa de controlo, uma vez que o próprio Arduino é alimentado a 5.0 V. A única alteração a ser efetuada neste aspeto é fornecer 5.0 V de alimentação a ambos os lados do M1.

[12] Siglas de *Master Out Slave In* e *Master In Slave Out*, respetivamente.
[13] Sigla de *Complementary Metal-Oxide Semiconductor*.

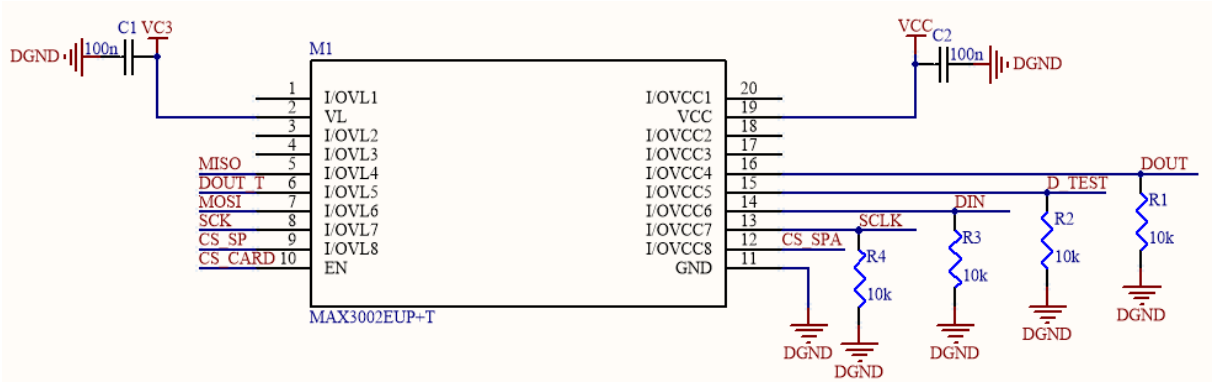


Figura 2.3 - Esquema elétrico da eletrónica associada ao conversor DC/DC MAX3002 [12].

Para efetuar o controlo dos componentes mencionados é necessário converter os dados transmitidos em série para parâmetros de controlo individual (paralelo). Esta operação é realizada com recurso a um expansor série (SPI)/paralelo (M2), ver Figura 2.4. Como o número de componentes a controlar nesta placa é inferior ao número de componentes da HV Remote, apenas se utiliza um expansor, enquanto na HV Remote são utilizados três. Como se pode observar na Figura 2.4, este expansor permite controlar o DAC (sinais DAC_CLR, DAC_SYNC e DAC_LDAC), o multiplexador analógico (sinais SEL_MUX, MUX_0, MUX_1, MUX_2 e MUX_3) e o ADC (sinal SEL_ADC).

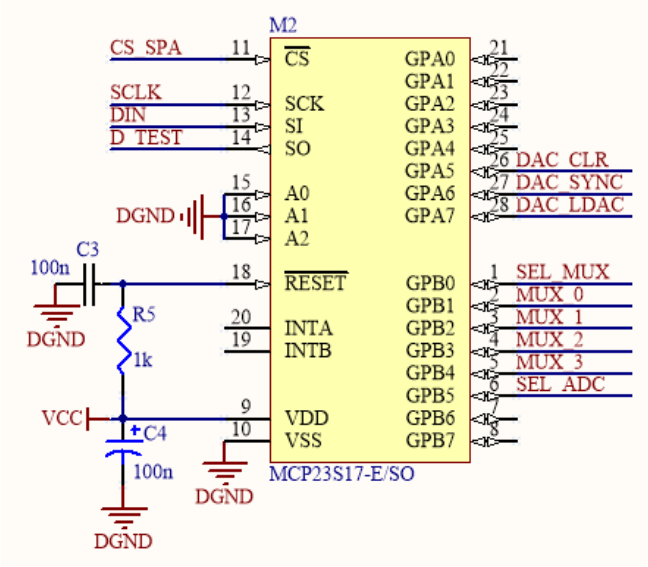


Figura 2.4 - Esquema elétrico da eletrónica associada ao expansor série (SPI)/paralelo MCP23S17 [13].

Os DACs fornecem o sinal analógico da tensão que permite a regulação individual da HV dos PMTs (tensão de entrada da malha de regulação) (Figura 2.1). Utiliza-se um DAC (Figura 2.5) de 8 canais, referido anteriormente como M3, o que permite efetuar o teste individual de cada um dos canais, com recurso a geradores digitais de ruído uniforme externos. Este teste permite calcular a não linearidade (parâmetros estáticos) e efetuar a análise de ruído destes conversores, ou seja, avaliar o seu desempenho na HV Remote. Para a sua realização, acrescentou-se ao circuito um amplificador operacional (Figura 2.6), que não existe na HV Remote, permitindo assim uma conexão direta entre o DAC e o ADC. Além disso, foi também acrescentado na placa um conector (Figura 2.6Figura 2.5) que permite selecionar um

de três valores de entrada possíveis do ADC: o valor de saída do DAC, o valor de saída do multiplexador ou um valor de tensão externo, VIN (para teste individual do ADC).

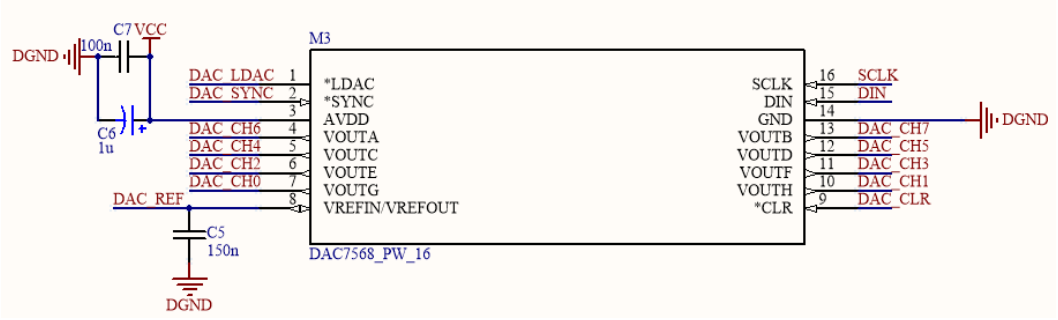


Figura 2.5 - Esquema elétrico da eletrónica associada ao DAC7568 [14].

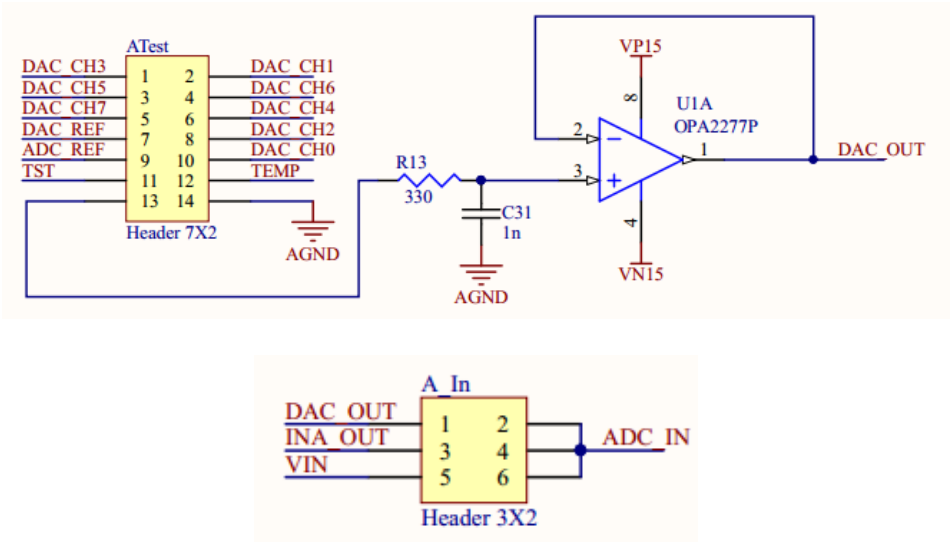


Figura 2.6 - Esquema elétrico da eletrónica associada ao amplificador operacional e conector adicionados para efetuar o teste do ADC e do DAC.

Para automatizar os testes do DAC, os seus sinais de saída são convertidos pelo ADC e enviados para o computador. Este teste acumula as não linearidades e erros do ADC, logo o ADC deve ser testado individualmente e os seus erros devem ser tomados em conta no teste do DAC. A tensão de entrada externa analógica, VIN, permite testar o desempenho do ADC escolhido recorrendo aos já mencionados geradores digitais de ruído. O ADC utilizado (Figura 2.7) também permite a leitura das tensões de referência dos conversores (DAC e ADC), da tensão de saída do sensor de temperatura (M8) (Figura 2.8), de uma tensão de referência de 1.23 V para teste (Figura 2.9), DZ1, e de uma tensão de entrada externa, VIN. Esta leitura é possível devido à utilização do multiplexador de 16 entradas, M4. A tensão de entrada externa analógica, VIN, permite também testar o desempenho do ADC escolhido recorrendo aos já mencionados geradores digitais de ruído. O ADC utiliza uma referência de tensão externa (REF02), anteriormente designada por M7 (Figura 2.7).

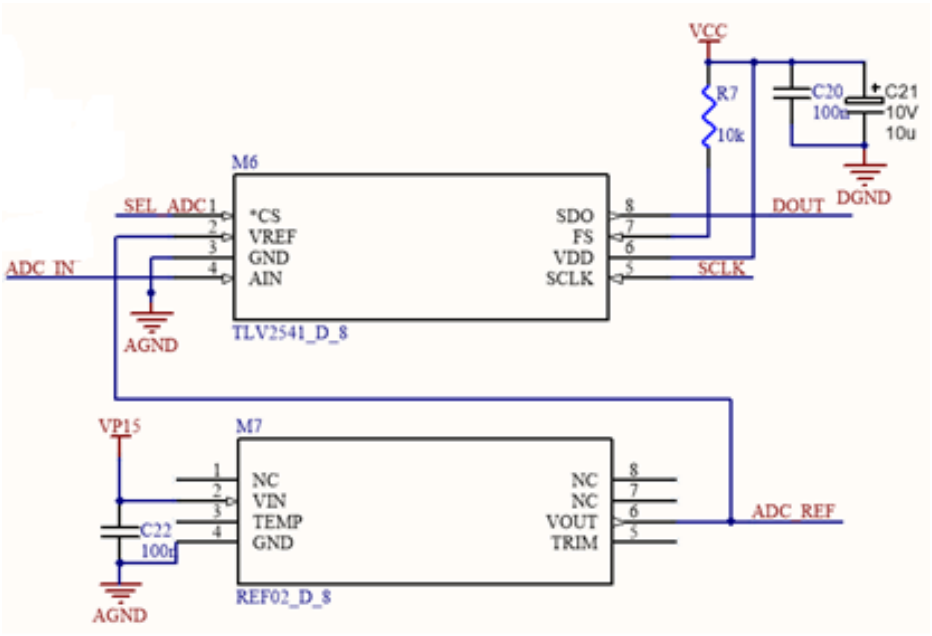


Figura 2.7 - Esquema elétrico da eletrónica associada ao ADC TLV2541 [15] e à sua referência de tensão REF02 [16].

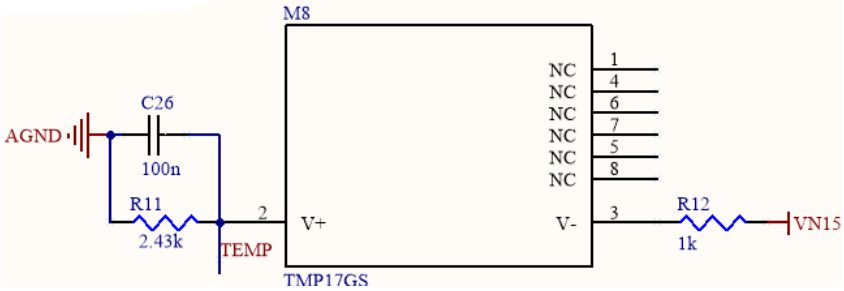


Figura 2.8 - Esquema elétrico da eletrónica associada ao sensor de temperatura TMP17 [17].

O sensor de temperatura, M8, é um transdutor de temperatura em circuito integrado que fornece uma corrente de saída proporcional à temperatura absoluta. Este sensor pode ser utilizado em aplicações com temperaturas entre os -40 ˚C e os 150 ˚C, permitindo saber a que temperatura se encontra o PCB em funcionamento, quando este for colocado num sistema em conjunto com outros PCBs, também em funcionamento, com pouco espaço entre eles. A relação entre a tensão e a temperatura é dada pela expressão (2.1).

$$Tensão\ (A) = (Temperatura\ (^{\circ}C) + 273) \times 10^{-6} \qquad (2.1)$$

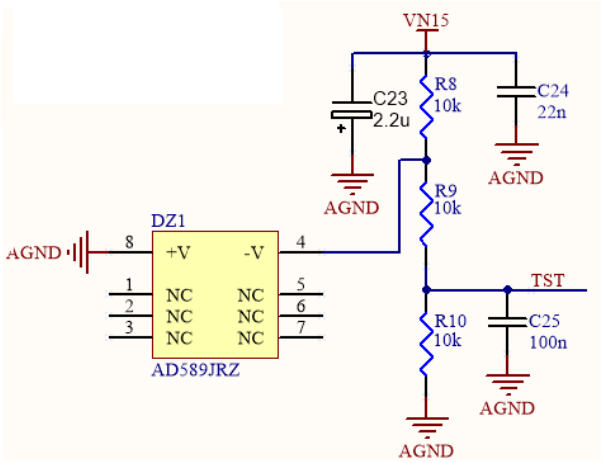


Figura 2.9 - Esquema elétrico da eletrónica associada à tensão de referência AD589 [18].

Como só é utilizado um ADC com entrada entre 0 – 5 V, utiliza-se um multiplexador analógico (Figura 2.10) com lógica associada à tabela que se encontra na Tabela 1 e um amplificador de instrumentação (Figura 2.10).

MUX_3	MUX_2	MUX_1	MUX_0	SEL_MUX	Canal Ativo
X	X	X	X	0	Nenhum
0	0	0	0	1	1
0	0	0	1	1	2
0	0	1	0	1	3
0	0	1	1	1	4
0	1	0	0	1	5
0	1	0	1	1	6
0	1	1	0	1	7
0	1	1	1	1	8
1	0	0	0	1	9
1	0	0	1	1	10
1	0	1	0	1	11
1	0	1	1	1	12
1	1	0	0	1	13
1	1	0	1	1	14
1	1	1	0	1	15
1	1	1	1	1	16

Tabela 1 - Lógica associada ao multiplexador MPC506.

Os valores lógicos (A0, A1, A2 e A3) enviados para o multiplexador permitem definir qual das suas 16 entradas será lida, sendo estes valores enviados com recurso a M2. Neste caso, são utilizadas apenas 13 entradas que correspondem aos canais de saída do DAC, à referência externa VIN, à referência de temperatura (Figura 2.8), à tensão de referência de 1.23 V (Figura 2.9) e às tensões de referência dos conversores. Na HV Remote, a HV fornecida aos PMTs é negativa. Então, o amplificador de instrumentação recebe o sinal proveniente do multiplexador e inverte o seu valor, uma vez que a tensão de entrada do ADC é positiva, e subtrai o ruído da massa analógica, para de seguida o transmitir à entrada do ADC. No caso da placa de controlo, o sinal não é invertido (não existem altas tensões), mas o componente foi mantido para que as condições de teste da placa sejam idênticas às condições de teste da HV Remote.

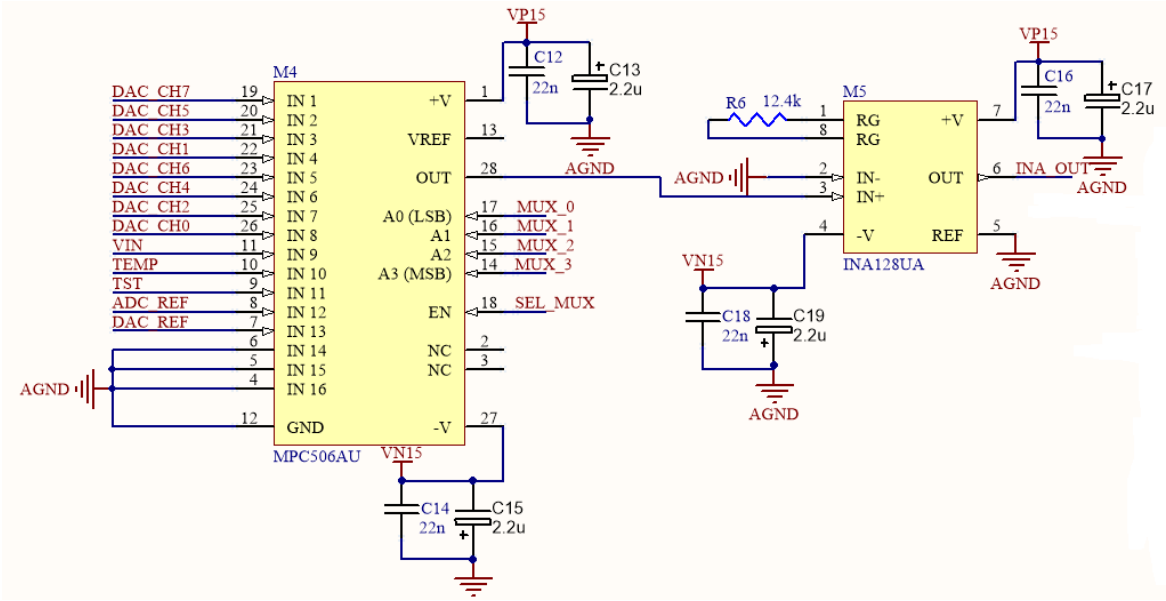


Figura 2.10 - Esquema elétrico da eletrónica associada ao multiplexador analógico MPC506 [19] e ao amplificador de instrumentação INA128 [20].

A carta HV Remote difere da placa de controlo por conter 3 expansores série (SPI)/paralelo, 3 DACs, 2 sensores de temperatura (em pontos opostos da placa) e os circuitos associados às altas tensões (uma de entrada externa e 24 de saída para os PMTs). Para replicar as suas condições de operação utilizam-se os mesmos valores de alimentação externa, +5 V e ±15 V, com exceção dos +3.3 V associados à ligação com a placa Tibbo, que não foi utilizada neste teste.

A placa de controlo, Figura 2.11, apenas tem dois planos, enquanto a HV Remote tem oito, pelo que foram inseridos dois planos de massa, um digital e analógico, separados, no plano inferior do PCB. Os componentes estão devidamente condensados nas respetivas zonas do PCB (digital e analógica), com exceção do DAC e do ADC, componentes mistos, que são atravessados por ambos os planos. A alimentação digital é de +5 V, enquanto a alimentação analógica é de ±15 V. A linha do sinal de relógio, fornecida através do Arduino por meio do protocolo SPI, também foi cuidadosamente colocada para evitar a indução de ruído nas linhas adjacentes.

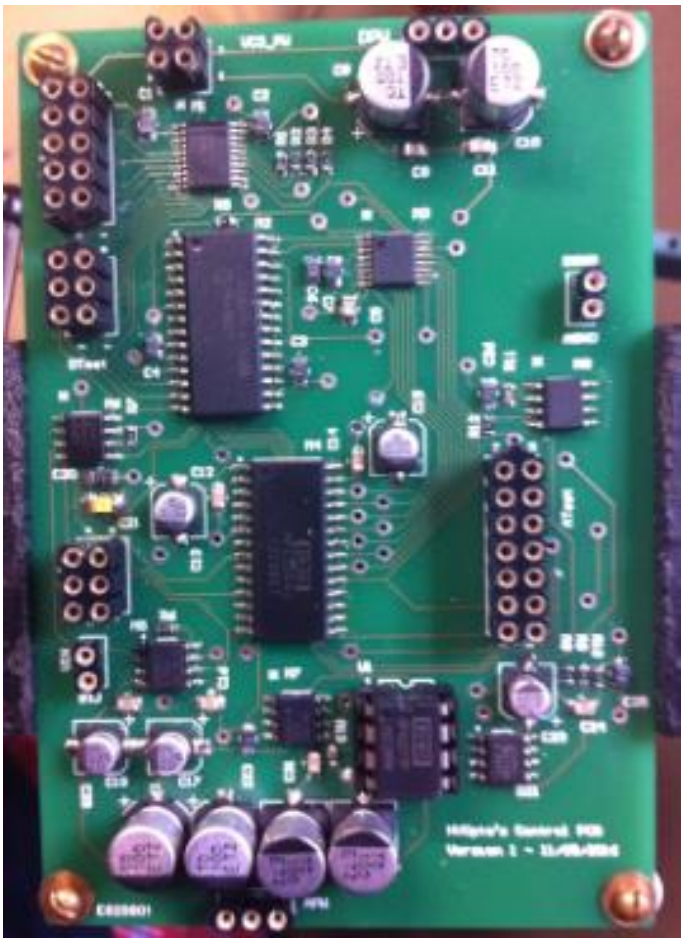


Figura 2.11 - Fotografia da 1ª versão da placa de controlo.

2.2 REFORMULAÇÃO DA PLACA DE CONTROLO

Como foi referido anteriormente, no decorrer dos testes iniciais à carta de controlo, foram encontrados alguns erros:

1. As massas do DAC não estavam corretamente atribuídas o que poderia impedir o seu funcionamento.
2. Existiam vários sinais que, consoante o teste a realizar, deviam ser visualizados no osciloscópio, mas não existiam pontos de teste que permitissem o acesso a esses sinais, tornando a sua observação impossível.
3. Verificou-se o aquecimento de um dos componentes, devido a uma má ligação de massas, não podendo ser assegurado o seu correto funcionamento a partir desse momento.

Além dos pontos referidos anteriormente, verificou-se a impossibilidade de comunicação quer com o DAC, quer com o ADC. Ao juntar todos estes fatores, decidiu-se que seria mais favorável o desenho de um novo protótipo. Este protótipo inclui todos os componentes descritos anteriormente, tendo sido acrescentados um novo DAC para comparação com o DAC7568 utilizado inicialmente e cujo funcionamento não foi o esperado, a respetiva referência de tensão e pinos de teste em todas as ligações existentes na placa.

O novo DAC MAX5725 (designado M9, Figura 2.12) é muito semelhante ao DAC7568. Apresenta oito canais de saída e ainda a possibilidade de utilização de uma referência de tensão interna ou externa. Tal como para o DAC7568, os seus canais serão testados individualmente com recurso a geradores de ruído

uniforme, existindo ainda a possibilidade de ligação à entrada do ADC (Figura 2.13). Ao acrescentar o novo DAC foi decidido deixar para o utilizador do circuito a utilização de uma referência de tensão externa ou a utilização da referência interna do DAC. Para tal, foi adicionada outra referência de tensão igual à utilizada para o ADC (Figura 2.14). Através do conector H9 (Figura 2.14) será possível utilizar a mesma referência para ambos os DACs ou utilizar a referência interna de ambos.

Além das já descritas alterações foi ainda adicionada uma opção para o pino LDAC de ambos os DACs. Este pino permite definir a forma de funcionamento dos canais dos DACs. Se estiver sempre ativo (com o valor binário '1'), os DACs estarão em funcionamento normal e os seus canais serão adaptados de acordo com os dados enviados para os seus pinos de entrada. Se estiver sempre desativado (com o valor binário '0'), os valores dos canais dos DACs serão adaptados de acordo com os valores guardados nos registos correspondentes a cada canal.

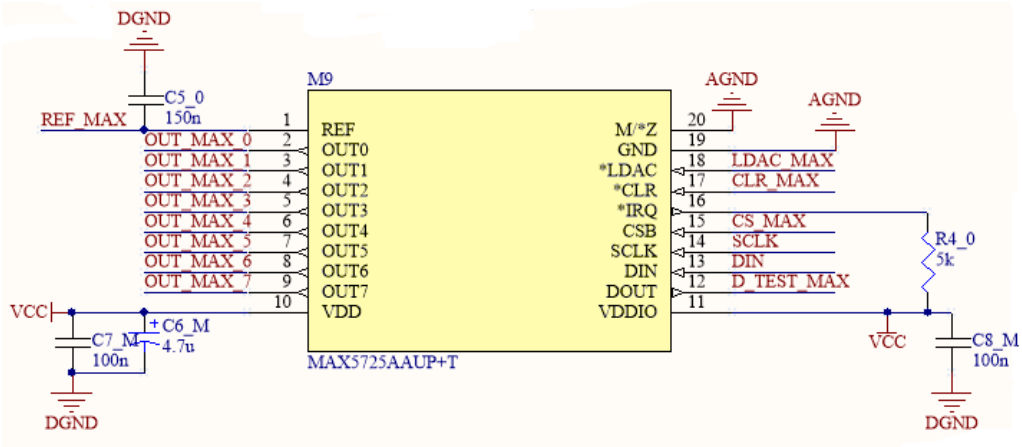


Figura 2.12 - Esquema elétrico associado à eletrónica do MAX5725 [21].

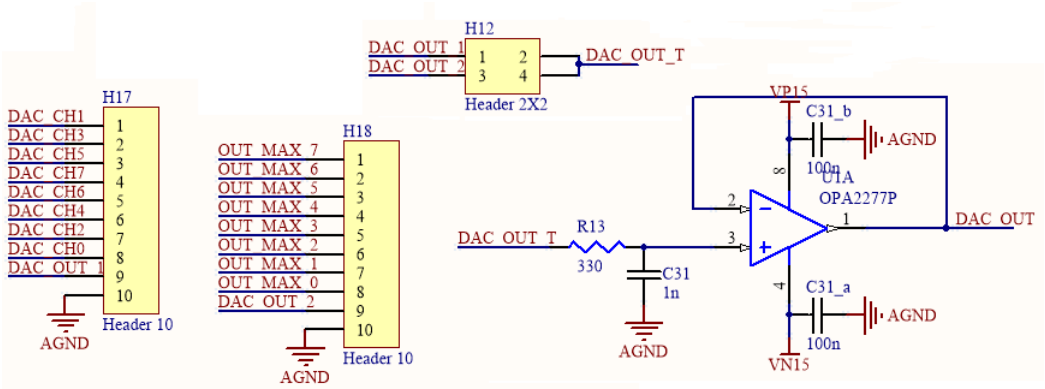


Figura 2.13 - Esquema elétrico da eletrónica associada ao amplificador operacional e conectores adicionados para efetuar o teste do ADC e do DACs.

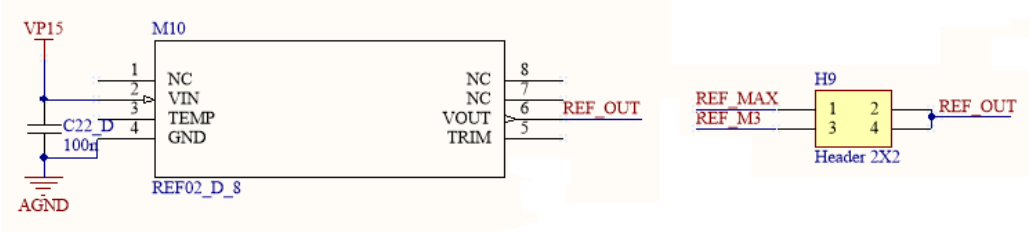


Figura 2.14 - Esquema elétrico associado à eletrónica da referência de tensão externa, REF02, dos DACs MAX5725 e DAC7568.

Ao expansor MPC23S17 e ao multiplexador MPC506 foram acrescentados os sinais necessários ao controlo (no expansor, Figura 2.15) e leitura dos canais do novo DAC (no multiplexador, Figura 2.16). O esquema elétrico da nova versão da placa de controlo encontra-se nos Anexos 8.5 e 8.6.

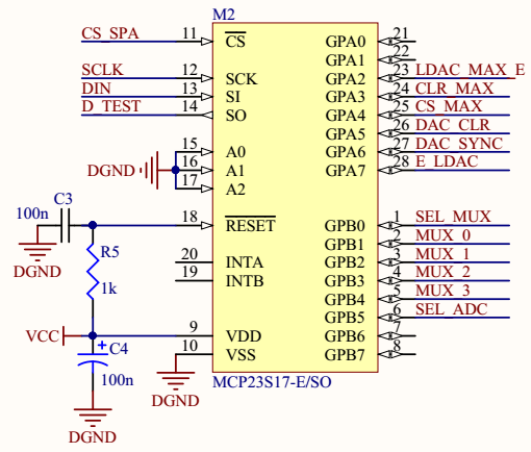


Figura 2.15 - Esquema elétrico associado à eletrónica do expansor MPC23S17.

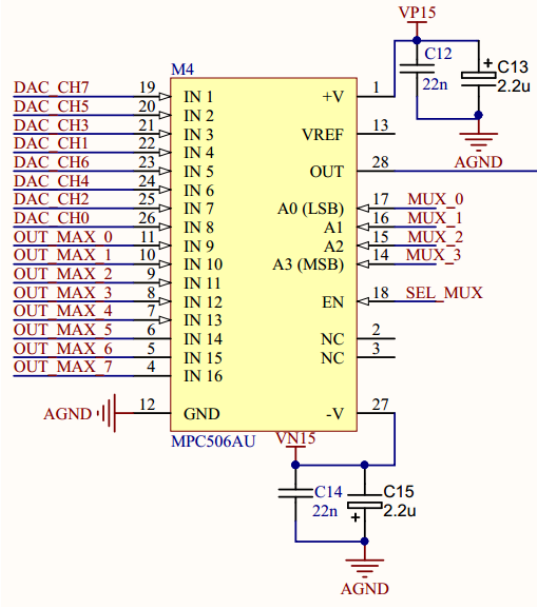


Figura 2.16 - Esquema elétrico associado ao multiplexador MPC506.

3 GERADORES DE RUÍDO DIGITAIS PARA O TESTE ESTÁTICO DE ADCS E DACS

Tantos os ADCs como os DACs funcionam como interface entre sinais analógicos e sinais digitais. Imagine-se que se quer aplicar uma tensão a uma resistência e se pretende visualizar essa tensão no ecrã de um computador. Por intermédio de um dispositivo com um ADC é possível transformar um valor analógico em digital e envia-lo para o computador e, por intermédio de um DAC incluído no *hardware* do computador, esse mesmo valor pode ser visualizado no ecrã do computador pelo seu utilizador. Existem milhares de dispositivos que recorrem a estes conversores para comunicarem entre si e com os seus utilizadores. O desempenho destes conversores vai determinar a exatidão de todo o sistema onde estão integrados, tornando-se o seu teste indispensável para a validação dos circuitos [22].

De seguida o funcionamento dos conversores é explorado, efetuando-se ainda uma descrição dos seus parâmetros de caracterização.

3.1 CONVERSORES ANALÓGICO – DIGITAL E DIGITAL – ANALÓGICO

Um ADC é um dispositivo que permite a transformação de um sinal de entrada analógico e contínuo numa sequência de dígitos binários discretos, sendo a função de transferência teórica de um ADC ideal uma reta. Na prática, a função de transferência ideal é uma característica uniforme em degrau [23]. Os degraus são projetados de forma a apresentarem transições tais que o ponto médio de cada degrau corresponde a um ponto na reta ideal (Figura 3.1). Cada código digital de saída representa um intervalo de valores analógicos da tensão de entrada. A diferença entre a tensão analógica de entrada e a correspondente tensão no centro de degrau mais próximo define-se como erro de quantificação. Como um conversor ADC apresenta um número finito de valores de saída (*bits*), mesmo um conversor ideal produz algum erro de quantificação em todas as amostras [23].

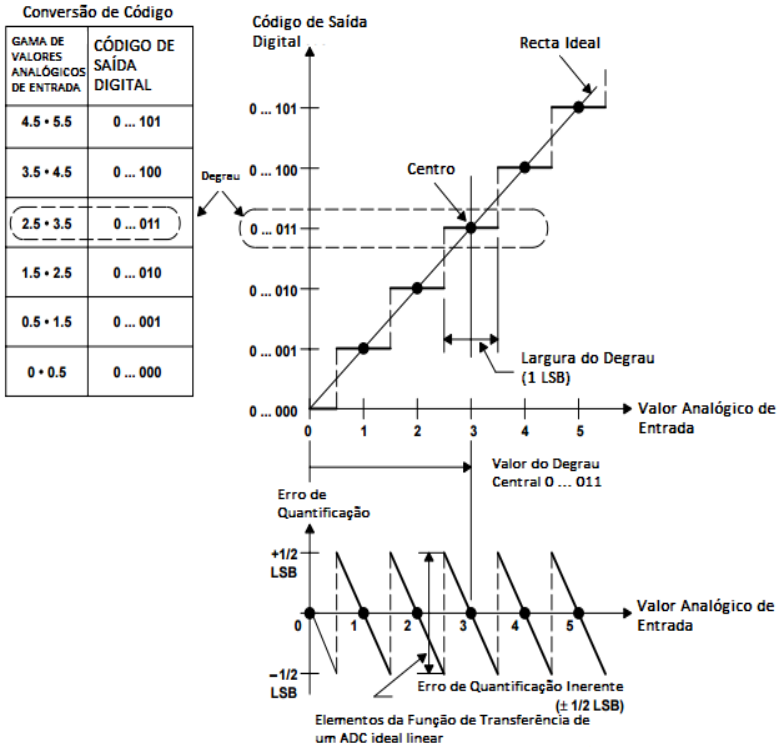


Figura 3.1 - Função de transferência de um ADC ideal.

A largura de um degrau, $L(i)$, define-se como 1 LSB[14]. Esta largura também é uma medição da resolução analógica do conversor, por definir o número de divisões da gama analógica completa. O valor ½ LSB representa uma quantidade analógica igual a metade da resolução analógica. A resolução de um ADC é expressa pelo número de *bits* no seu código de saída digital. Um ADC com resolução de N *bits* apresenta 2^N códigos digitais possíveis, ou seja, 2^N-1 níveis de degrau. O primeiro degrau (zero) e o último apenas apresentam metade da largura total, quando existe compensação do erro de quantificação.

Um DAC é um dispositivo que apresenta um número limitado de códigos de entrada digitais que são convertidos em valores de saída analógicos e discretos. Tal como no caso de um ADC ideal, a sua função de transferência teórica corresponde a uma reta verificando-se, na realidade, que a função de transferência corresponde a uma série de pontos coincidentes com essa reta (Figura 3.2).

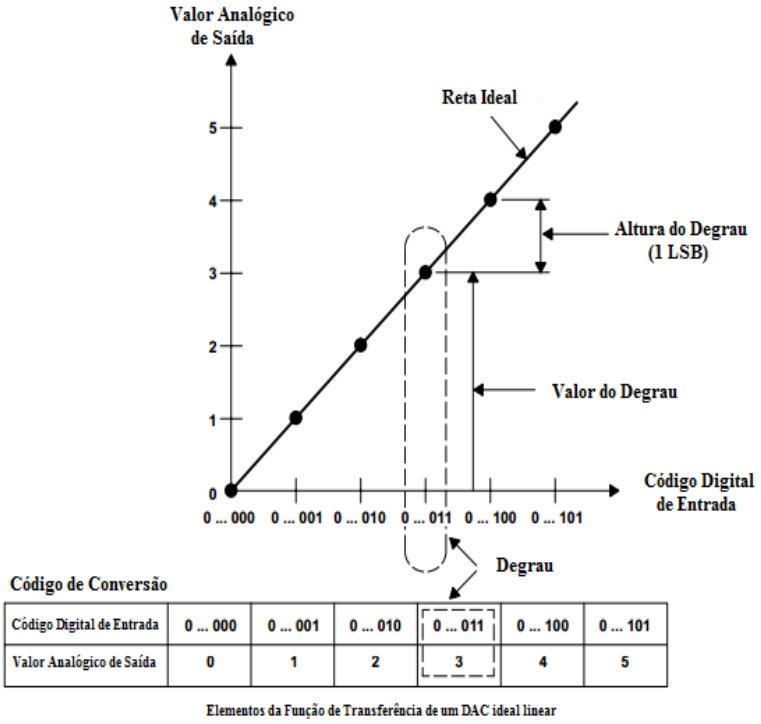


Figura 3.2 - Função de transferência de um DAC ideal.

A altura de um degrau, $L(i)$, define-se como 1 LSB. Esta altura também é uma medição da resolução analógica do conversor, por definir o número de divisões da gama analógica completa. Além disso, a altura de um degrau define-se como a diferença entre dois valores de saída analógicos sucessivos. Tal como para o ADC, o valor ½ LSB representa uma quantidade analógica igual a metade da resolução analógica. Um DAC de N bits apresenta 2^N níveis de saída possíveis.

As fronteiras entre os níveis possíveis são designadas por tensões de transição $V_t(i)$, com $i = 1, 2, \dots, 2^N - 1$.

Formalmente, a largura (ADCs) ou a altura (DACs) de um nível são representadas por

$$L(i) = V_t(i+1) - V_t(i) \tag{3.1}$$

[14] Sigla de *Least Significant Bit.*

Seja V_{in} o sinal analógico de entrada, tem-se o seguinte valor para a tensão quantificada do ADC:

$$V'_{in} = V_{in} \pm V_x \tag{3.2}$$

Com $-\frac{1}{2}V_{LSB} \leq V_x \leq \frac{1}{2}V_{LSB}$ e $V_{LSB} = \frac{V_{REF}}{2^N - 1}$, sendo V_{REF} o limite da gama do sinal quer do ADC quer do DAC.

3.2 PARÂMETROS CARACTERÍSTICOS DOS CONVERSORES

A relevância dos parâmetros característicos dos conversores depende da aplicação onde estes são utilizados. Para o teste geral dos conversores interessa explorar os seus parâmetros estáticos, mais propriamente a tensão de desvio, o erro de ganho e a não-linearidade diferencial e integral. Existem ainda parâmetros dinâmicos que se encontram para além do âmbito desta dissertação. Além dos parâmetros estáticos descrevem-se, na próxima secção, algumas características dos conversores. Apesar de ser feita distinção entre algumas definições, é importante referir que as definições dos parâmetros estáticos são válidas tanto para os ADCs como para os DACs, sendo a única diferença o domínio dos sinais de entrada e de saída [24]. O único parâmetro inerente unicamente ao ADC é o erro de quantificação.

3.2.1 Parâmetros Estáticos

Estes parâmetros definem caraterísticas dos conversores que podem ser testadas a baixas velocidades ou com tensões constantes [25]. Para cada parâmetro, o teste dos conversores determina se este passa ou falha o teste de acordo com os valores de saída do conversor.

Exatidão

A exatidão define-se como a diferença entre a característica de transferência ideal e a observada, tendo em conta a tensão de desvio e os erros de ganho e linearidade. Se a exatidão for superior à resolução do conversor, significa que a sua função de transferência é obtida com uma elevada exatidão.

Erro de Quantificação

Tal como já foi referido, este parâmetro parâmetro apenas diz respeito ao ADC. Todas as tensões de entrada contidas num intervalo $\pm\frac{1}{2}V_{LSB}$ do centro de um degrau na função de transferência dizem respeito a um mesmo código digital. Pode existir uma pequena diferença entre o centro do degrau e a respetiva tensão de entrada, devido à sua quantificação. O erro criado por esta quantificação comporta-se como ruído. O estudo deste ruído é feito através da sua análise estatística considerando uma aproximação estocástica, ou seja, supondo que o sinal de entrada apresenta uma variação tão rápida que o erro de quantificação pode ser considerado uma variável aleatória uniformemente distribuída. Pode-se assim calcular o valor *rms* (*root mean square)* esperado desta quantificação de ruído. Este erro está representado na Figura 3.1, sendo dado por:

$$E\{\epsilon^2\} = \frac{1}{V_{LSB}}\int_{-\frac{1}{2}V_{LSB}}^{+\frac{1}{2}V_{LSB}} \epsilon^2 d\epsilon = \frac{1}{V_{LSB}}\left[\frac{\epsilon^3}{3}\right]_{-\frac{1}{2}V_{LSB}}^{+\frac{1}{2}V_{LSB}} = \frac{V_{LSB}^2}{12} \tag{3.3}$$

Resolução

Num ADC, a resolução de um conversor define-se como o seu número de *bits*. Uma resolução de N *bits* significa que o conversor distingue entre $2^N - 1$ intervalos de valores analógicos diferentes. A resolução define o tamanho do LSB, ou seja, determina a gama dinâmica, a largura dos níveis e o erro de quantificação.

É ainda importante a definição do número efetivo de *bits*, ENOB, ao qual se encontra associada a definição de resolução efetiva. Este parâmetro define-se como

$$ENOB = \log_2 \frac{V_{REF}}{NC_{rms}\sqrt{12}} \tag{3.4}$$

Sendo NC_{rms} o valor *rms* do erro de conversão. Este erro depende de parâmetros do ADC e da taxa de repetição do sinal analógico de entrada. Num conversor ideal, o ENOB é igual a N.

No caso do DAC, como já foi referido anteriormente, a resolução define-se como a alteração da saída analógica que se verifica quando existe a variação de 1 LSB na entrada digital, sendo dada pela seguinte expressão:

$$Resolução = \frac{V_{REF}}{2^N} \tag{3.5}$$

Sendo 2^N o valor dos níveis de saída possíveis.

Tensão de Desvio, V_{OS}

No caso do ADC, quando a tensão de entrada corresponde a um valor nulo, o código digital de saída também deve ser '0'. Se isto não se verificar, o valor médio da tensão de entrada que produz o código '0' é a tensão de desvio.

Para um DAC, a tensão de desvio corresponde ao valor analógico de saída quando a sua entrada digital se encontra a '0'.

Erro de Ganho, G

Este erro define-se como a diferença entre a curva de transferência ideal e a real, após a correção da tensão de desvio.

Não-Linearidade Diferencial (DNL)

A DNL define-se, no caso do ADC, como o desvio das larguras dos níveis de transição, ou no caso do DAC, como o desvio das alturas dos níveis de transição, em comparação com a largura ideal de 1 LSB.

$$DNL(i) = \frac{GL(i) - V_{LSB}}{V_{LSB}} \tag{3.6}$$

Na expressão anterior, o produto $GL(i)$ é a largura/altura de um nível após a correção do erro de ganho, subtraída ao valor da largura/altura ideal e dividida por esse mesmo valor. Idealmente, a DNL seria zero.

Não-Linearidade Integral (INL)

A INL define-se como a soma desde a primeira conversão até à atual da não-linearidade diferencial de cada nível de quantificação:

$$INL(i) = \sum_{j=1}^{i} DNL(i) \tag{3.7}$$

Por exemplo, se a soma da DNL até um nível particular for 1 LSB, isso significa que o total de larguras/alturas do nível até esse ponto é 1 LSB superior à soma do total de larguras/alturas ideais do código, ou seja, a tensão de entrada será convertida um código abaixo da conversão ideal.

A INL representa a diferença entre as tensões de transição ideal e atual, após a correção do erro de ganho e da tensão de desvio, sendo formalmente definida pela expressão:

$$INL(i) = \frac{GV_t(i)+V_{OS}-V_{ideal}(i)}{V_{LSB}} \tag{3.8}$$

Sendo, G o erro de ganho, $V_t(i)$ a tensão de transição do nível i, V_{OS} a tensão de desvio, $V_{ideal}(i)$ a tensão ideal do nível i e V_{LSB} a tensão correspondente a 1 LSB.

Códigos Não Atribuídos

Um ADC e um DAC podem apresentar códigos não atribuídos (intervalos de quantificação com um valor nulo de amostras) se os valores máximos dos respetivos módulos da DNL forem superiores a 1 LSB ou se os da INL forem superiores a ½ LSB.

3.3 TESTE AUTOMÁTICO DOS CONVERSORES

Como mencionado anteriormente, um conversor pode ser caracterizado consoante os seus parâmetros estáticos ou dinâmicos, consistindo essa caracterização na determinação quantitativa das suas características. Os parâmetros estáticos descrevem o desempenho fundamental do conversor para sinais de entrada contínuos ou de variação lenta. Os parâmetros dinâmicos caracterizam o comportamento do conversor para sinais de entrada discretos ou de rápida variação no tempo.

Os testes estáticos são utilizados para determinar as tensões dos níveis de transição e permitem a determinação dos parâmetros estáticos. Estes testes são muito lentos, tornando-se uma limitação para conversores de alta resolução. Por outro lado, os parâmetros do conversor, bem como as condições de operação, podem variar durante o teste.

Existem vários métodos para uma rápida caracterização de conversores: o método do histograma e análises no domínio da frequência ou no domínio temporal. O método do histograma permite uma rápida caracterização estática do conversor [26] e é o mais sensível a erros de não-linearidade diferencial. Este método é também o mais apropriado para caracterizações em circuito integrado. As análises no domínio da frequência consistem na aplicação de um sinal sinusoidal como estímulo e na análise do espectro resultante. As análises do domínio temporal também utilizam um sinal sinusoidal como estímulo adquirindo-se um certo número de amostras para definir a sinusoide que melhor se adapta a essas

amostras. Estas duas últimas análises permitem a determinação do SNR[15] e da distorção harmónica, sendo mais sensíveis a erros de não-linearidade integral [26].

O teste de conversores de alta resolução com sinais de natureza determinística[16] apresenta alguns problemas que serão descritos no ponto seguinte. O foco deste trabalho vai estar centrado apenas no método do histograma, sendo a sua descrição efetuada na secção seguinte. Relembra-se que, apesar de a descrição ser efetuada para ADCs este método pode ser analogamente aplicado a DACs.

3.3.1 Método do Histograma

O princípio de funcionamento do método do histograma baseia-se em colocar um sinal na entrada do conversor, com distribuição de amplitude conhecida. O sinal de estímulo deve varrer toda a gama de tensões de entrada do conversor e permitir a representação no histograma do número de vezes que cada código de saída do ADC ocorre [26]. Ao considerar-se um sinal de estímulo ideal, em que todos os níveis de tensão apresentam a mesma probabilidade de ocorrência, um ADC ideal apresentará um histograma com um número igual de amostras em cada nível de quantificação. Comparações entre o número de amostras obtido experimentalmente em cada nível de quantificação e o número esperado num conversor ideal permitem a determinação da largura de cada nível de quantificação do conversor e da respetiva tensão de transição. É assim possível determinar a característica de transferência do conversor em teste, o que permite o cálculo de outras características do conversor. Este método permite a determinação da característica de transferência do conversor para diferentes frequências do sinal de estímulo, sendo possível quantificar a degradação do seu desempenho com a frequência. O método do histograma deve ser complementado por uma análise no domínio da frequência em caso de erros de linearidade elevados no ADC em teste, ou de um elevado número de códigos não atribuídos [26].

Este método baseia-se na análise estatística dos sinais de entrada e saída do ADC, apresentando, no entanto, algumas desvantagens como a necessidade de um estímulo com uma função de densidade de probabilidade conhecida em toda a largura de banda do ADC e a necessidade de se ter em consideração a sobreposição de ruído parasita ao estímulo. Os dois tipos de sinais frequentemente utilizados como estímulos no método do histograma são: os sinais sinusoidais e os sinais em dente de serra ou triangulares. No entanto, ambos os sinais apresentam uma série de desvantagens que influenciam os resultados obtidos com o teste, nomeadamente, a sua natureza determinística. Estes sinais, devido à sua periodicidade, apresentam a necessidade de tanto a frequência do sinal como as frequências de amostragem serem tais que deva ser evitada a redundância de informação. Além disto, necessitam que a exatidão do gerador do sinal seja superior à exatidão expectável para o ADC em teste. Outra desvantagem, a nível de aplicações de banda larga, como as telecomunicações, é a largura de banda do teste ser definida pelo gerador do sinal de estímulo e a necessidade de sequências de teste longas.

Alternativamente, os geradores de ruído são uma boa opção como sinal de estímulo. Estes sinais apresentam várias vantagens:

1. O ruído é mais fácil de gerar (em circuito integrado) que uma onda sinusoidal ou triangular, sendo o seu gerador completamente descrito pela distribuição do sinal de saída;
2. Se o ruído gerado seguir uma distribuição Gaussiana, qualquer ruído adicional apenas soma a sua variância à do ruído gerado;

[15] Sigla de *Signal-To-Noise Ratio*.
[16] Um sinal determinístico é um sinal cujos valores, em qualquer instante, podem ser conhecidos *à priori.*

3. O ruído branco é um sinal de banda larga, ou seja, com uma única caracterização é possível estimar os parâmetros estáticos do ADC numa banda de frequência elevada.

Estes sinais estocásticos apresentam várias distribuições, sendo as mais importantes, neste âmbito, as distribuições uniforme e a Gaussiana.

A distribuição uniforme apresenta uma boa razão entre o número de amostras necessário para se efetuar o teste e o respetivo intervalo de incerteza. É, no entanto, difícil de gerar em circuitos integrados, especialmente a altas frequências, sendo sensível ao ruído térmico ou de interferência, devido à distribuição normal que o ruído normalmente apresenta. A distribuição Gaussiana pode ser associada a ruído térmico ou de interferência de banda larga, podendo ser gerado com relativa facilidade, especialmente a altas frequências [27]. A utilização do ruído uniforme é mais simples e menos dispendiosa, em termos de espaço, quando este é gerado em FPGA[17].

A utilização de geradores de ruído como estímulos para o teste automático em conversores analógico-digitais apresenta assim uma série de vantagens face aos estímulos tradicionais.

3.4 GERADORES DE RUÍDO

O ruído pode ser gerado no domínio analógico e no digital [28]. Os métodos analógicos tornam possível a geração de valores aleatórios verdadeiros, ou seja, não existe correlação entre os valores de ruído gerados, mas são sensíveis ao ambiente operacional. Apresentam consumos superiores aos digitais, sendo uma das origens de alto consumo a utilização comum de amplificadores operacionais, e apresentam uma baixa gama dinâmica. Os valores gerados por métodos digitais podem ser pseudoaleatórios ou aleatórios. No caso dos pseudoaleatórios, o período de geração pode ser grande o suficiente para que as sequências apenas de repitam ao fim de um intervalo muito elevado. Métodos de geração de ruído baseados em algoritmos específicos ou em caos permitem a geração de números aleatórios verdadeiros. A maior desvantagem dos geradores digitais face aos analógicos é a discretização do sinal, que exige a utilização de um conversor digital-analógico e um filtro passa-baixo para obter o sinal analógico final. Esta questão não se coloca no teste de DACs, devido às suas entradas digitais. O foco deste trabalho é a descrição de algoritmos baseados em sequências numéricas pseudoaleatórias e em caos. Far-se-á primeiramente uma breve descrição de algumas características do ruído.

3.4.1 Ruído

A análise temporal do ruído utiliza grandezas para a sua caracterização, como o valor *rms*, a razão sinal-ruído, o número de ruído e o fator de ruído [26]. Neste trabalho, apenas serão utilizados os valores *rms* de fontes de tensão e de corrente de ruído que se definem por:

$$V_n = \left[\frac{1}{T}\int_0^T v_n^2(t)dt\right]^{\frac{1}{2}} \tag{3.9}$$

$$I_n = \left[\frac{1}{T}\int_0^T i_n^2(t)dt\right]^{\frac{1}{2}} \tag{3.10}$$

Sendo T um intervalo de tempo que quanto maior, mais exata será a estimativa efetuada.

[17] Sigla de *Field-Programmable Gate Array*.

A análise em frequência é também útil no estudo e caracterização do ruído. A densidade espectral de ruído, V_n^2 ou I_n^2, define-se como sendo a média da potência de ruído normalizada para uma largura de banda de 1 Hz. V_n^2 é uma função de valores reais positivos, cujo valor quadrático médio de ruído aleatório é nulo para qualquer intervalo de frequência. O valor quadrático médio total pode ser obtido através da seguinte integração:

$$V_n^2 = \int_0^\infty V_n^2(f)df \tag{3.11}$$

3.4.1.1 Tipos de Ruído

Para além do ruído de interferência, entre o sistema de teste do ADC e o mundo exterior ou entre elementos do próprio sistema (ruído este que pode ou não ser aleatório), existem três fontes importantes de ruído a considerar. A primeira fonte é o ruído térmico, também designado por ruído de *Johnson* ou de *Nyquist* [29]. Este tipo de ruído é provocado pela vibração aleatória e termicamente excitada dos portadores de carga de um condutor. Todos os condutores, com temperatura superior ao zero absoluto, possuem eletrões em movimento aleatório (movimento Browniano), sendo este movimento dependente da temperatura.

A densidade espectral do ruído é independente da frequência, verificando-se que o ruído térmico é um ruído branco[18] [26]. A tensão rms do ruído térmico, para uma resistência, é dada por:

$$V_n = \sqrt{4kTR\Delta f} \tag{3.12}$$

Sendo k a constante de Boltzmann, T a temperatura absoluta e Δf a largura de banda do ruído.

O ruído térmico apresenta uma distribuição de amplitudes gaussiana e não se distingue de outros ruídos com a mesma distribuição se a caracterização for efetuada a apenas uma temperatura.

A segunda fonte considerada é o ruído de *Shot*, também denominado por ruído granulado. Este ruído está presente em a díodos e transístores, podendo a sua origem ser explicada pelos fenómenos associados a uma junção *p-n*. É possível modular a passagem de cada portador, através da junção, como um fenómeno aleatório dependente da energia e direção de deslocamento do portador. Ao aplicar-se uma tensão na junção, vai ser estabelecida uma corrente, devida à passagem de um grande número de impulsos de corrente aleatórios e independentes. A componente flutuante desta corrente é o ruído *Shot*. O ruído quadrático médio da corrente de ruído resultante é dado por:

$$I_n = \sqrt{2qI_D\Delta f} \tag{3.13}$$

Sendo q a carga do eletrão, I_D o valor médio de um elevado número de impulsos de corrente aleatórios e independentes e Δf a largura de banda. A validade desta expressão mantém-se para valores de frequência próximos de $\frac{1}{\tau}$, sendo τ o tempo necessário para os portadores atravessarem a região de depleção da junção *p-n*. De acordo com a expressão anterior, a densidade espectral de corrente de ruído é constante na frequência, sendo por isso este ruído um ruído branco.

A terceira fonte de ruído é o ruído *Flicker* ou ruído de tremulação. A baixas frequências, o ruído de um componente semicondutor, frequentemente, excede o valor esperado nas contribuições do ruído térmico e de *Shot*. Este tipo de ruído apresenta uma densidade espectral inversamente proporcional à frequência,

[18] Ruído branco é um sinal aleatório que apresenta uma densidade espectral constante.

sendo as suas características variáveis de dispositivo para dispositivo, sugerindo que este ruído se encontra associado a detalhes da estrutura do dispositivo. As origens deste ruído são variadas e não são conhecidas em detalhe [30].

As referidas principais fontes de ruído em circuitos eletrónicos afetam o sistema de teste dos conversores na sua globalidade. Podem ser utilizadas nos geradores de ruído analógicos para o teste dos conversores e em alguns geradores digitais aleatórios. Os geradores de ruído digitais pseudoaleatórios são baseados na geração de sequências numéricas que são descritas de seguida, mais propriamente com recurso a geradores de ruído gaussianos e de ruído uniforme. Os geradores de ruído digitais aleatórios são baseados em circuitos caóticos.

3.4.2 Geradores de Ruído Gaussianos

Os métodos para geração de números aleatórios gaussianos utilizam princípios matemáticos básicos, que por norma envolvem a transformação de números aleatórios uniformes. Os geradores de números aleatórios gaussianos são divididos em quatro categorias básicas: inversão da função cumulativa de distribuição (CDF[19]) e métodos de transformação, rejeição e recursão [31].

Uma distribuição gaussiana com média zero e desvio-padrão unitário é conhecida como distribuição normal e a sua função densidade de probabilidade (*pdf*[20]) é dada por:

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \tag{3.14}$$

O gráfico desta *pdf* não indica, diretamente, a probabilidade de ocorrência de uma gama de valores de x. A integração da expressão anterior fornece a função cumulativa de densidade.

$$\Phi(x) = \int_{-\infty}^{x} \phi(x)dx = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right] \tag{3.15}$$

Esta expressão dá a probabilidade de uma amostra aleatória da distribuição gaussiana apresentar um valor menor que x, podendo ser utilizada para calcular a probabilidade de valores que ocorrem num certo intervalo. Esta função não é de forma fechada[21].

3.4.2.1 Inversão CDF

A inversão CDF funciona através da utilização de um número aleatório y, gerado por um gerador de números aleatórios uniformes que produz números aleatórios com distribuição uniforme no intervalo contínuo (0,1), excluindo 0 e 1 [31]. Este número aleatório y é utilizado para a geração de um número aleatório gaussiano x através da inversão $x = \Phi^{-1}(y)$.

$$\Phi^{-1}(y) = 2 \times \sqrt{2}\,\text{erf}^{-1}(y-1) \tag{3.16}$$

A função ϕ associa números gaussianos com uma probabilidade entre 0 e 1, enquanto a função Φ^{-1} associa números entre 0 e 1 a números gaussianos. O facto de $\Phi^{-1}(y)$ não ser de forma fechada para a

[19] Sigla de *Cumulative Distribution Function*.
[20] Sigla de *probability density function.*
[21] Uma função diz-se de forma fechada se é possível ser expressa analiticamente em termos de um número finito de funções bem conhecidas.

distribuição gaussiana torna necessário o uso de aproximações, o que pode afetar a qualidade dos números aleatórios resultantes. A inversão gaussiana CDF utiliza aproximações polinomiais.

3.4.2.2 Transformada de Box-Muller

A transformada de Box-Muller gera um par de variáveis aleatórias que obedecem à distribuição normal, a partir de um par de variáveis aleatórias uniformes [32]. Sejam U_1 e U_2 variáveis aleatórias independentes, pertencentes à mesma função de densidade no intervalo (0,1) . Consideram-se as seguintes variáveis aleatórias:

$$X_1 = (-2\ln U_1)^{1/2} \cos\ (2\pi U_2) \tag{3.17}$$

$$X_2 = (-2\ln U_1)^{1/2} \sin\ (2\pi U_2) \tag{3.18}$$

O par (X_1, X_2) representa um par de variáveis aleatórias independentes com distribuição normal, com média nula e variância unitária.

3.4.2.3 Teorema do Limite Central

O teorema do limite central afirma que a soma de variáveis aleatórias independentes e igualmente distribuídas converge na direção da distribuição normal com o aumento do número de termos da soma. Então, a soma de n variáveis uniformemente distribuídas, $s = u_1 + u_2 + \cdots + u_n$, pode ser considerada uma aproximação da distribuição gaussiana. Os valores obtidos devem ser escalados para que a sua média seja nula e a sua variância unitária. Então, a distribuição

$$g = \sqrt{\frac{12}{n}}\left(s - \frac{n}{2}\right) \tag{3.19}$$

Converge para a distribuição normal desejada, sendo obtida da subtração da média $\mu_s = \frac{n}{2}$ a s, e da sua divisão pelo desvio padrão $\sigma_s = \sqrt{\frac{12}{n}}$.

3.4.2.4 Aproximação Linear por Partes

A distribuição gaussiana é decomposta num conjunto de k componentes básicas com distribuições triangulares, $t_1 \ldots t_k$, cada uma com um tamanho de $2w$, centradas em em $c_i = w\left(\frac{k+1}{2} - i\right)$, com uma probabilidade q_i associada. O espaçamento regular $(2w)$ significa que cada triângulo se sobrepõe com os triângulos imediatamente à sua esquerda e à sua direita. A soma de sobreposições cria uma aproximação linear por partes a uma *pdf* gaussiana. As saídas são criadas por primeiramente se escolher um dos triângulos e posteriormente gerar um número aleatório a partir da distribuição do triângulo selecionado. Os triângulos são selecionados através do método *Alias*[22] para amostragem de uma distribuição discreta, utilizando uma entrada uniforme. As distribuições triangulares são geradas através da soma de duas variáveis de entrada uniformes, propriamente escaladas [31].

[22] Método criado por A. J. Walker. Devolve valores inteiros entre 1 e n, de acordo com uma distribuição de probabilidade aleatória, p_i.

3.4.2.5 Métodos de Rejeição

Seja $y = f(x)$ uma função com um integral finito, C o conjunto de pontos (x, y) debaixo da curva e Z a área finita que é um superconjunto[23] de C. São extraídos uniformemente pontos aleatórios (x, y) de Z, até que $(x, y) \in C$ e x é devolvido como número aleatório.

Um destes métodos é o **método polar de Marsaglia** que se relaciona com a transformada de Box-Muller mas é mais rápido e mais exato [33]. No entanto, pode necessitar de vários conjuntos de números uniformes até encontrar valores que não são rejeitados. Os números aleatórios gerados são dados pelas seguintes equações [34]:

$$Z_0 = x\sqrt{\frac{-2\ln(x^2+y^2)}{(x^2+y^2)}} \tag{3.20}$$

$$Z_1 = y\sqrt{\frac{-2\ln(x^2+y^2)}{(x^2+y^2)}} \tag{3.21}$$

Se $(x^2 + y^2) > 1$, o conjunto de pontos é rejeitado e são gerados novos números aleatórios uniformes.

Outro dos métodos utilizado é o **método de razão de números uniformes** que pode ser ajustado para uma grande variedade de distribuições. Seja X uma variável aleatória com uma *pdf* $f(x) = \frac{g(x)}{\int g(x)dx}$, sendo $g(x)$ uma função positiva integrável, com suporte[24] (x_0, x_1). Se (V, U) é uniformemente distribuído em $A = \{(v, u): 0 < u \leq \sqrt{g\left(\frac{v}{u}\right)}, x_0 < \frac{v}{u} < x_1\}$, então $X = V/U$ apresenta uma densidade de probabilidade proporcional a g. Os parâmetros deste método são definidos consoante a área A escolhida [35].

Finalmente, o **método de Ziggurat** utiliza uma curva a delimitar a parte positiva da *pdf* gaussiana, que é escolhida como a união de *n* secções, R_i $(1 \leq i \leq n)$, constituídas por $(n - 1)$ rectângulos, e pela zona da cauda da *pdf*. Estas zonas são escolhidas de forma a terem áreas iguais, v, e os seus cantos direitos são denotados por x_i. Todos os rectângulos, menos um, podem ser divididos em duas regiões: outro retângulo, delimitado à direita por x_{i-1}, que está completamente dentro da *pdf* , e à direita uma região de forma semelhante a um triângulo, que incluí regiões acima e abaixo da *pdf*. O retângulo limitado por x_1 consiste apenas numa região com forma semelhante a triângulo. Sempre que é gerado um número aleatório, uma das secções é escolhida aleatoriamente. É gerada uma amostra uniforme, que é avaliada para ver se está dentro de um retângulo que está completamente dentro da *pdf*, dentro da secção escolhida. Se isso acontecer, a amostra é a amostra de saída gaussiana [36].

3.4.2.6 Método Recursivo

Wallace propõe um gerador de números aleatórios que assenta na propriedade de combinações lineares de números aleatórios gaussianos apresentarem distribuições gaussianas. Existem inicialmente $N = KL$ números aleatórios independentes com distribuição normal, normalizados para que o seu valor médio ao quadrado seja unitário. Em L passos de transformação, os K números são tratados como um vetor X (constituído pelos K números) e transformados em K novos números com $X' = AX$, sendo A

[23] Todos os pontos de C estão contidos em Z.
[24] Menor subconjunto fechado do domínio, onde a função não é nula.

uma matriz ortogonal. Se os valores originais de K tiverem distribuição gaussiana, também os novos valores terão [31].

3.4.3 Geradores de Ruído Uniforme

O ruído uniforme corresponde a sequências de números $u_0, u_1, u_2, \dots, u_n$ de forma a que cada u_n satisfaça $0 \leq u_n \leq 1$ e, que cada u_n tenha igual probabilidade e que seja estatisticamente independente dos restantes valores, ou seja, $u_k \neq u_n$. As técnicas mais comuns produzem sequências de inteiros uniformemente distribuídos $x_0, x_1, \dots$ tal que $0 \leq x_n < B$, sendo B o limite superior da sequência, e existe um inteiro positivo P mínimo, que representa o período, para o qual a relação $x_{n+P} = x_n$ é satisfeita para todos os n. A sequência de [0, 1) variáveis uniformes torna-se $\frac{x_0}{B}, \frac{x_1}{B}, \dots, \frac{x_{P-1}}{B}$. Esta sequência não é uniformemente distribuída devido à existência de um número de elementos finitos. No entanto, se P e B forem grandes o suficiente, a sequência é praticamente uniforme. Existem cinco técnicas comuns para a criação de uma sequência de números uniformemente distribuídos: geradores congruentes, geradores de registo de deslocamento, geradores de atraso de Fibonacci, geradores de combinação [37] e geradores de Mersenne Twister.

3.4.3.1 Geradores Congruentes

Considere-se um conjunto finito X e uma função $f: X \to X$, que transforma elementos de X noutros elementos de X. Dado um valor inicial $x \in X$, a sequência gerada é o tipo $x, f(x), f^2(x), f^3(x), \dots$.

O conjunto finito X é um conjunto de resíduos reduzidos de um módulo m e $f(x) = (ax + b)\ mod\ m$. Então, para um elemento inicial $x_0 \in X$, a sequência gerada é $x_0, x_1, \dots$, com $x_{n+1} = (ax_n + b)\ mod\ m$, com $n = 0,1,2, \dots$, sendo m um inteiro com valor elevado, que determina o período do gerador. A sequência $\{x_i/m\}$ é uma sequência de números aleatórios uniformes. Se $b = 0$, o gerador diz-se multiplicativo [38].

3.4.3.2 Geradores de Registo de Deslocamento

O conjunto finito X é um conjunto de $1 \times k$ vectores binários, $x = (b_1, b_2, \dots, b_k)$ e a função f é uma transformação linear, $f(x) = xT$, sendo T uma matriz binária $k \times k$ com módulo aritmético 2[25]. Com um vetor binário x inicial, a sequência é $x, xT, xT^2, xT^3, \dots$, sendo a matriz T escolhida de forma a que o período seja longo e que a multiplicação por T seja uma implementação relativamente rápida [38].

3.4.3.3 Geradores de Atraso de Fibonacci

A fórmula geral de um gerador de atraso de Fibonacci apresenta-se como [39]:

$$LF[r, s, m, \circ: x_t\{0, \dots, r-1\}] \tag{3.22}$$

Sendo $r > s > 0$ os atrasos, $\circ$ uma operação binária, m a base utilizada e $x_t\{0, \dots, r-1\}$ uma sequência de r valores iniciais. Para $n \geq r$, a sequência caracteriza-se por:

[25] O resultado desta operação é 1, se o resultado for ímpar, e 0, se o resultado por par.

$$x_n = x_{n-r} \circ x_{n-s} \pmod{2^m} \tag{3.23}$$

As operações binárias utilizadas são, por norma, a adição, a subtração, a multiplicação de módulo m e a operação XOR $\oplus$, se m for uma potência de 2. Normalmente, $m = 2^N$, sendo N o número de *bits* do sistema. Este tipo de geradores devem ser inicializados aleatoriamente, com recurso a outro gerador de números aleatórios. Os números obtidos com este gerador dependem da inicialização do mesmo.

3.4.3.4 Geradores de Mersenne Twister

O algoritmo de Mersenne Twister produz uma sequência de números pseudoaleatórios uniformes entre 0 e 2^w-1, sendo w o número de *bits* correspondente ao tamanho da sequência gerada. Este algoritmo baseia-se na seguinte equação de recorrência [40]:

$$x_{k+n} \coloneqq x_{k+m} \oplus \left(x_k^u \middle| x_{k+1}^l\right) A \tag{3.24}$$

Sendo o inteiro n o grau da recorrência, m um inteiro definido como $1 \leq m \leq n$ e A a matriz quadrada constante, de dimensão w, com entradas pertencentes a $\{0,1\}$. x_k^u representa os $w - r$ *bits* superiores de x_k e x_{k+1}^l representa os r *bits* inferiores de x_{k+1}, sendo r definido como $0 \leq r \leq w - 1$. A expressão $\left(x_k^u \middle| x_{k+1}^l\right)$ denota a concatenação de dois elementos, sendo o resultado desta operação multiplicado pela matriz A.

As amostras são produzidas de forma a seguirem uma distribuição aleatória uniforme. A implementação mais utilizada designa-se por *MTT19937* e produz uma sequência com um período de $2^{19937} - 1$ números. O tamanho do período denomina-se por número primo de Mersenne. A geração de um número x através da utilização de um elemento y do vector de palavras é definida segundo o seguinte procedimento:

$$x \leftarrow y \oplus (\, y \gg 11) \tag{3.25}$$

$$x \leftarrow x \oplus (\, (x \ll 7) \odot b) \tag{3.26}$$

$$x \leftarrow x \oplus (\, (x \ll 15) \odot c) \tag{3.27}$$

$$x \leftarrow c \oplus (\, y \gg 18) \tag{3.28}$$

Sendo $\oplus$ o operador XOR, $\odot$ o operador AND e OR e $\gg, \ll$ os operadores de deslocamento lógico à direita e à esquerda, respectivamente. Para que o valor de y seja actualizado, sejam y_1 e y_{397} os elementos $(i + 1)$ e $(i + 397)$ do vector, executam-se as seguintes operações:

$$y \leftarrow (\, y \odot mm) || (y_1 \odot ll) \tag{3.29}$$

$$y \leftarrow y_{397} \oplus (\, y \gg 1) \tag{3.30}$$

$if\ (y\ mod\ 2 == 1)$

$$x \leftarrow c \oplus (\, y \gg 18) \tag{3.31}$$

Os parâmetros a, b, c, ll e mm são escolhidos de acordo com a descrição realizada por Matsumoto e Nishimura[26].

3.4.3.5 Geradores de Combinação

Consideram-se combinações aditivas e subtrativas. Consideram-se também duas sequências de inteiros aleatórios $\{x_n\}$ e $\{y_n\}$, com limites B_x e B_y, respectivamente. Seja $m \geq \max(B_x, B_y)$. Define-se a sequência de combinação das sequências como:

$$z_n = x_n \pm y_n \; mod \; m \tag{3.32}$$

As variáveis uniformes [0, 1) seriam $\frac{z_n}{m}$, para a operação $+$ ou $-$, [37].

3.4.4 Geradores de Ruído Digitais Baseados em Circuitos Caóticos

Vários geradores caóticos analógicos têm sido implementados utilizando elementos discretos e amplificadores operacionais, bem como sistemas completamente baseados em tecnologia MOS[27] ou sistemas integrados. Este tipo de geradores são sensíveis a variações de condições operacionais e de temperatura [41]. Como tal, condições iniciais deste tipo de sistemas não podem ser implementadas com precisão. Além disso, a implementação do circuito analógico, por norma, requere uma área relativamente grande de implementação. Existem algumas propostas de implementações digitais de circuitos caóticos que são mencionadas de seguida.

Um gerador de números aleatórios baseado em caos diferencial digital onde as condições iniciais e estados do sistema são guardados em registos[28] em vez de condensadores, como nos analógicos, é descrito em [41]. O sistema é caracterizado por largas margens de ruído[29] e um elevado nível de confiança. Este gerador é baseado num sistema caótico descrito como

$$-\dddot{X} = \ddot{X} + B\dot{X} + X \tag{3.33}$$

Onde o elemento não linear se define como,

$$B(\dot{X}) = \begin{cases} \alpha, & \dot{X} \geq 1 \\ 0, & \dot{X} < 1 \end{cases} \tag{3.34}$$

Geradores caóticos baseados em diferenças também podem ser implementados digitalmente através da solução numérica das suas equações diferenciais. Estas soluções podem ser implementadas através da utilização de um módulo de transferência de registos onde as variáveis associadas às soluções são implementadas como registos e cada uma das equações é realizada como uma unidade de lógica combinada.

Outro método disponível é um gerador de ruído que utiliza um fenómeno designado por *jitter*, resultante do ruído térmico proveniente de osciladores em anel [42]. O ruído a cada estado do oscilador provoca

[26] Referência ao artigo *"Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator"*.
[27] Sigla de *Metal-Oxide Semiconductors.*
[28] Registos são dispositivos de armazenamento de dados que consistem num grupo de básculas em cascata utilizadas para guardar *bits* de informação.
[29] Margem de ruído é a quantidade de ruído para o qual um circuito ainda apresenta o modo de funcionamento esperado. Este valor é definido para que valores positivos assegurem operações próprias do circuito e para que valores negativos resultem em falhas de funcionamento.

desvios no período de oscilação. A incerteza no período é o chamado *jitter*. Utilizando esta incerteza para realizar o sinal de relógio de uma báscula D, é possível obter uma saída aleatória.

Dos métodos apresentados ao longo deste capítulo foram escolhidos dois geradores uniformes, *Mersenne Twister* e o Congruente Multiplicativo para efetuar os testes dos conversores. Estes testes e respetivos resultados serão discutidos no Capítulo 5.

4 DESCRIÇÃO DA INTERFACE DE UTILIZADOR E COMUNICAÇÃO COM A PLACA DE CONTROLO

Para testar a carta de controlo da HV Remote foi desenvolvida uma interface de utilizador em *Python*. Esta interface permite a comunicação entre um computador e a carta, por intermédio do protocolo Série/SPI. As informações são enviadas através do protocolo série do computador para um Arduino Uno e, posteriormente, enviadas por SPI do Arduino para a carta. Mais propriamente, estava previsto que a interface permitisse efetuar o teste do expansor série/paralelo, do DAC e do ADC, bem como o teste de desempenho da carta, recebendo os vários valores de tensões analógicas referidos no segundo capítulo (as tensões de referência do DAC e do ADC, a temperatura de referência e a tensão de referência de 1.24 V).

Procede-se de seguida à descrição dos protocolos de comunicação utilizados bem como da interface desenvolvida, dos testes efetuados e dos resultados obtidos.

4.1 PROTOCOLOS DE COMUNICAÇÃO

Para o desenvolvimento desta interface foram considerados dois protocolos de comunicação: série e SPI.

O protocolo SPI foi desenvolvido pela *Motorola*, tendo sido apresentado como a ligação externa a um microcontrolador que permitia a ligação dos seus periféricos a quatro cabos (as quatro linhas de sinal mencionadas no segundo capítulo) [43]. As linhas de sinal definem-se da seguinte forma:

1. Um sinal de relógio (SCLK[30]) é enviado do *master* (dispositivo que controla) para todos os *slaves* (dispositivos controlados), sendo os sinais SPI síncronos com este sinal.
2. Um *slave select* (SS) que pode assumir vários valores (um de cada vez), consoante o dispositivo a controlar, e que permite que o *master* selecione o dispositivo com que quer comunicar.
3. Uma linha de dados do *master* para o *slave*, denominada por *Master Out-Slave In* (MOSI).
4. Uma linha de dados do *slave* para o *master*, denominada por *Master In-Slave Out* (MISO).

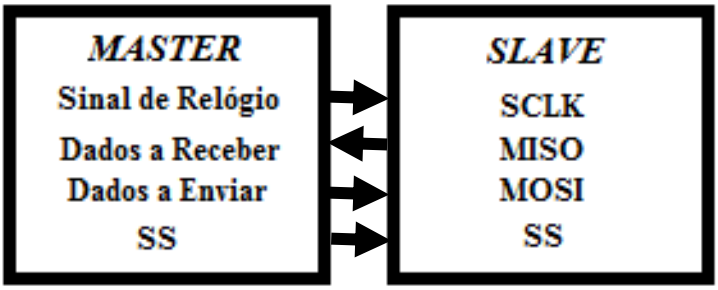


Figura 4.1- Correspondência entre as linhas de dados do master e do slave no protocolo SPI.

Este protocolo apenas envolve um *master* que controla a comunicação com todos os *slaves*. Quando o *master* pretende enviar ou pedir dados de um *slave*, o SS do dispositivo a controlar é selecionado através da transição do estado lógico '1' para o estado lógico '0' e o SCLK é ativado a uma frequência suportada tanto pelo *master* como pelo *slave*. O *master* envia informação pela linha MOSI e recebe pela MISO.

[30] Sigla de *signal clock*.

Existem quatro modos de comunicação possíveis, definidos com recurso a dois parâmetros: a polaridade do relógio (CPOL de *Clock Polarity*) e a fase do relógio (CPHA de *Clock Phase*) [44]. Os valores destes parâmetros correspondem a estados lógicos digitais e estão representados na Figura 4.2. Os estados correspondem a '1' ou 'HIGH' e '0' ou 'LOW'.

Modo	Polaridade (CPOL)	Fase (CPHA)
0		
1		
2		
3		

Figura 4.2 - Modos de comunicação do protocolo SPI.

Um par de dispositivos *master/slave* tem que utilizar os mesmos parâmetros (frequência, CPOL e CPHA) para que a comunicação entre eles seja possível. Se existem múltiplos *slaves* com diferentes configurações, o *master* deve ser reconfigurado sempre que comunica com um dispositivo diferente [43].

O protocolo série é utilizado na comunicação entre o computador e o Arduino. Os Arduinos são construídos incluindo diversos sistemas para comunicação em série. Este protocolo é automaticamente controlado pelo Arduino e o programador apenas tem que se preocupar com os dados a enviar e a receber [45]. A comunicação série pode ser classificada de duas formas:

1. Síncrona, ou seja, os dispositivos em comunicação utilizam o mesmo sinal de relógio e apresentam tempos de comunicação sincronizados.
2. Assíncrona, ou seja, os dispositivos têm sinais de relógio diferentes que são ativados pelo sinal de saída de estados anteriores.

Neste caso específico, o Arduino comunica com o computador através do módulo UART[31] que é assíncrono. A comunicação série assíncrona apresenta mecanismos que asseguram transferências de dados robustas e sem erros. Existem quatro mecanismos [45]:

1. Os *bits* de sincronização representam dois ou três *bits* transferidos com cada pacote de dados. Estes *bits* designam-se por inicial e final, representando o início e o fim de um pacote de dados. O *bit* inicial é único, inserindo-se após uma transição do estado lógico '1' para o estado lógico '0', enquanto o *bit* final pode ser único ou podem ser enviados dois *bits*, que estão associados à transição para o estado lógico '1' e posterior permanência nesse valor.
2. Os *bits* de dados representam a quantidade de dados num pacote, sendo que o seu tamanho pode variar entre 5 e 9 *bits*, apesar de os pacotes padrão apresentarem 8 *bits*.
3. Os *bits* de paridade podem ou não existir, de acordo com a escolha do utilizador, que também define o seu valor. O *bit* de paridade assume o valor '0' se o número de '1' no pacote de dados é par, para o caso ímpar o raciocínio é análogo.
4. A taxa de transmissão é utilizada para denotar o número de *bits* transferidos por segundo.

[31] Sigla de *Universal Asynchronous Receiver Transmitter*.

Das descrições efetuadas de ambos os protocolos utilizados, retira-se que a maior preocupação a cargo do programador é, de facto, o tratamento dos dados a enviar e a receber, uma vez que ambos os protocolos se encontram bem definidos. Mais especificamente, a comunicação com o Arduino é efetua-se através de uma ligação de cabo USB entre o computador e a porta USB do Arduino, sendo apenas necessário uma linha de código (referida mais adiante) na área de desenvolvimento do Arduino para iniciar a comunicação.

4.2 Interface da Carta de Controlo

O desenvolvimento desta interface englobou tanto o nível funcional e de interação com o utilizador como a sua parte gráfica. A escolha de utilização do *Python* como linguagem de programação para desenvolver esta interface deveu-se, parcialmente, ao facto do grande leque de pacotes disponíveis especificamente para desenvolvimento gráfico. Além disso, esta linguagem fornece uma grande variedade de bibliotecas que permitem a comunicação com vários tipos de dispositivos como, por exemplo, a biblioteca *serial* para a comunicação entre a interface e o Arduino ou a *pyvisa* para a comunicação com instrumentos da *National Instruments*. Para a parte gráfica da interface, escolheu-se o pacote *Tkinter*.

A interface apresenta três funções principais: o teste dos expansores série/paralelo (mencionados no segundo capítulo), o teste do DAC e do ADC e, por fim, o teste de desempenho da placa. Estas opções são apresentadas num painel principal (Figura 4.3) onde o utilizador pode carregar no botão pretendido. Esta ação despoleta o aparecimento de uma nova janela, onde é possível realizar diferentes tarefas, consoante a opção escolhida.

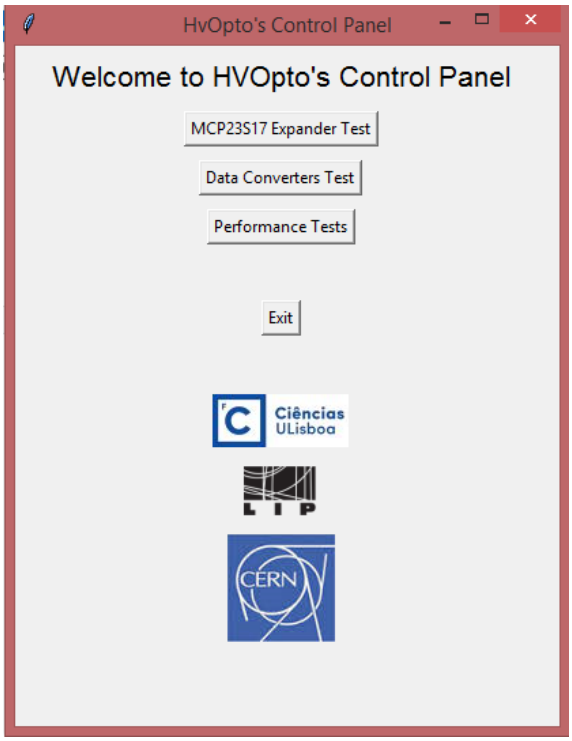


Figura 4.3 - Painel principal da interface em Python.

4.2.1 Expansor Série/Paralelo

A comunicação com o expansor série/paralelo foi facilitada pela existência de uma classe específica para o mesmo e preparada para ser utilizada com o Arduino. Esta classe foi desenvolvida por Cort Buffington em Março de 2012 e pode ser consultada em [46]. Tal como o próprio autor refere, a implementação desta classe fornece uma interface que, de certa forma, replica as funções de entrada e saída do próprio Arduino. Para o seu funcionamento, é necessário importar a classe *SPI*. São descritos de seguida, na Tabela 2, os métodos utilizados para a realização da interface.

MCP()	
Descrição	Inicialização do expansor como um objeto.
Sintaxe	*MCP nome_do_objecto(endereço)*
Parâmetros	*nome_do_objeto:* nome arbitrário do objeto. *endereço:* endereço correspondente aos pinos A0, A1 e A2 do expansor, entre 0 e 7.
Retorno	Nenhum
pinMode()	
Descrição	Configuração dos portos do expansor como entrada ou saída.
Sintaxe	*nome_do_objeto.pinMode(modo)*
Parâmetros	*nome_do_objeto:* nome dado ao objeto quando foi criado. *modo:* palavra com o formato '0BXXXXXXXXXXXXXXXX' que indica o modo (entrada ou saída) de cada um dos 16 pinos de entrada/saída do expansor; o valor '1' coloca o pino como entrada e o valor '0' coloca o pino como saída.
Retorno	Nenhum
digitalWrite()	
Descrição	Escrever o código digital '1' ou '0' para os pinos digitais de entrada do expansor.
Sintaxe	*nome_do_objeto.digitalWrite(pino,valor)*
Parâmetros	*nome_do_objeto:* nome dado ao objeto quando foi criado. *pino*: o número do pino (1 a 16) ao qual o valor será atribuído *modo:* se um pino for especificado, um dos códigos digitais, '1' ou '0' será escrito para esse pino; se nenhum pino for especificado, o *valor* deve ser uma palavra de 16 *bit* que especifica quais os pinos de entrada e saída do expansor.
Retorno	Nenhum
digitalRead()	
Descrição	Ler o valor dos pinos de entrada
Sintaxe	*nome_do_objeto.digitalRead(pino)*
Parâmetros	*nome_do_objeto:* nome dado ao objeto quando foi criado. *pino:* o número do pino (1 a 16) cujo valor será lido; se nenhum pino for especificado, serão lidos todos os pinos configurados como entrada.

Tabela 2- Descrição dos métodos inerentes à classe do expansor.

4.2.2 Ligação entre a Interface de Utilizador e o Arduino

Como foi referido numa das secções anteriores, a biblioteca *serial* permite configurar a ligação entre a interface desenvolvida e o Arduino. Na parte da interface, deve-se importar a biblioteca *serial* e declarar um objeto, a ser utilizado em todo o código, que faz referência à conexão. Esta declaração é feita com recurso a (4.1).

$$ser = serial.Serial('COM3', 9600, timeout = 0) \quad (4.1)$$

Neste caso, o objeto declarado designa-se por *ser*, enquanto *serial* representa a utilização da biblioteca série importada e *Serial* é um dos métodos/funções disponíveis nessa biblioteca. Dos vários parâmetros que a biblioteca *Serial* pode receber, apenas interessam três: o porto utilizado ('COM3'), a taxa de transmissão (9600 *bits* por segundo) e o tempo em que a ligação vai expirar (neste caso, a ligação só expira quando o utilizador a fechar explicitamente).

Para enviar dados, utiliza-se (4.2). Para garantir que os dados passados a esta função chegam intactos ao Arduino, devem ser codificados com recurso à função *encode()*. Esta função permite codificar os dados enviados de forma a que o Arduino os consiga interpretar.

$$ser.write(dados.encode()) \quad (4.2)$$

Além de serem enviados dados da interface para o Arduino, também serão enviados dados do Arduino para a interface. Os dados recebidos do Arduino são lidos com recurso a (4.3). Tal como se procedeu à codificação dos dados enviados, também se deve proceder à descodificação dos dados recebidos.

$$ser.readline().decode() \quad (4.3)$$

Por último, deve-se utilizar a função $ser.close()$ para fechar a conexão série entre a interface e o Arduino.

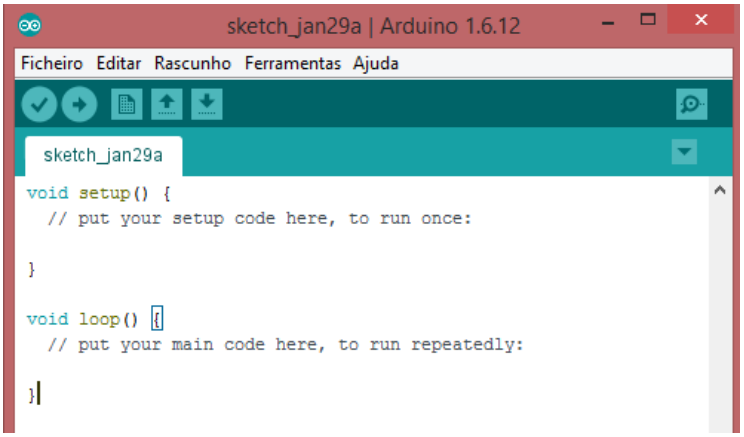


Figura 4.4 - Interface do Arduino.

Da parte do Arduino, também é necessário existir algum processamento de dados. O código base do Arduino divide-se em duas funções: *setup* e *loop* (Figura 4.4). O programador pode desenvolver o número de funções que pretender, mas para o objetivo deste trabalho tal não foi necessário, tendo-se apenas utilizado estas funções base. A função *setup* é onde são definidas as configurações do Arduino e apenas é executada no início do programa. Para que os dados enviados da interface em *python* sejam reencaminhados para o Arduino é necessário iniciar a conexão série, tal como se fez na interface. Este processo realiza-se com recurso a (4.4).

$$Serial.begin(9600) \quad (4.4)$$

A função *loop* é continuamente executada. Quando chega ao fim do código, volta ao início e processa o código novamente. Nesta função, verifica-se se a conexão em série está ou não disponível (4.5). Caso esteja, os dados enviados pela interface para o Arduino são lidos, depositados numa variável

previamente declarada (4.6) e são posteriormente enviados de volta para a interface (4.7), onde são recebidos com recurso a (4.3).

$$if(Serial.available()) \tag{4.5}$$

$$char\ inByte = Serial.read() \tag{4.6}$$

$$Serial.print(inByte) \tag{4.7}$$

A descrição efetuada nesta secção resume os aspetos básicos da comunicação entre a interface de utilizador e o Arduino. Ao longo da descrição da interface será explicado todo o raciocínio efetuado para tornar possível a transmissão de dados.

4.2.3 Interface do Expansores Série/Paralelo

Um dos elementos fundamentais da HV Remote é o expansor. Através dele é possível controlar todo o circuito e, como tal, é necessário testar o seu desempenho.

O MCP23S17 é, como já foi referido, um expansor série/paralelo de 16-*bit*, ou seja, é constituído por 16 pinos bidirecionais, que constituem os portos A e B, cada um com 8 pinos, sendo estes portos independentes entre si e os pinos podem ser configurados como pinos de entrada ou saída, permitindo comunicações de alta velocidade (até 10 MHz) por SPI.

Na janela correspondente ao teste do expansor Figura 4.5 é possível testar se os dados enviados para o expansor são, de facto, os recebidos, sendo utilizada a conexão entre a interface e o Arduino descrita na secção anterior.

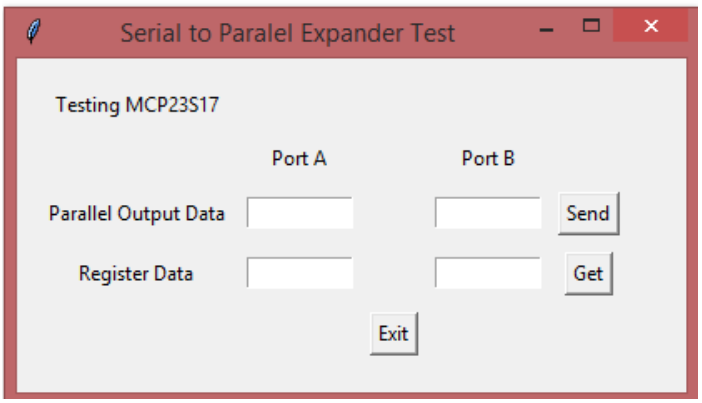


Figura 4.5 - Janela de teste do expansor.

O utilizador insere na interface a palavra de 16-*bit* a enviar para o expansor, sendo ambos os portos configurados como entrada, separada em duas palavras de 8-*bit* cada, sendo que cada palavra se refere a um porto diferente do expansor (A ou B). Os dados inseridos pelo utilizador são enviados para o Arduino e guardados num ficheiro, para posterior análise.

No Arduino, o primeiro passo é incluir a classe *SPI* e *MCP23S17* bem como declarar o expansor.

```
#include <SPI.h>
#include <MCP23S17.h>

MCP expansor(0,10);
void setup() {
  Serial.begin(9600);
  expansor.begin();
  pinMode(8,OUTPUT);
  digitalWrite(8, HIGH); //Enable do MAX3002
  expansor.pinMode(0B1111111111111111);
  delay(500);
}
```

Figura 4.6 - Declaração de bibliotecas, declaração do expansor e função setup no ambiente de desenvolvimento do Arduino.

Neste bloco da interface, a função *setup*, para além da inicialização da porta série, permite a configuração do expansor. Os métodos utilizados estão descritos na Tabela 2. É necessário utilizar o método $digitalWrite(pino, valor)$ para ativar o conversor DC/DC MAX3002. Caso o conversor não seja ativado, o resto do circuito não irá funcionar pois está dependente dos dados transmitidos através deste componente. No método $expansor.pinMode(X)$ define-se que se vai escrever para todos os pinos dos portos A e B do expansor.

A cada ciclo da função *loop* vai ser lido um dos *bits* da sequência de 16-*bit* enviados pela interface. Utiliza-se um contador, iterado no fim de cada ciclo da função, para garantir que os *bits* serão transmitidos para os devidos pinos, ou seja, cada iteração do *loop* corresponde à escrita para um pino específico. No início da função, atribui-se à variável *inByte* o valor do *bit* recebido através da conexão série. Através de uma sequência *if…else if*, faz-se a distribuição dos *bits*: os primeiros oito serão enviados para o porto A, enquanto os seguintes serão enviados para o porto B. Para garantir que o valor enviado para cada pino é o correto, faz-se a comparação entre o *bit* recebido, sob a forma de palavra, com a palavra "1" ou "0", utilizando-se posteriormente o método $expansor.digitalWrite(pino, valor)$ para enviar os dados para o expansor. Como o contador é inicializado com o valor zero, o valor do pino onde se quer escrever corresponde a *count + 1* (Figura 4.7, penúltima linha de código). Quando todos os *bits* são recebidos e transmitidos ao expansor, é efetuada a leitura dos valores registados em cada pino, sendo essa leitura enviada para a interface de utilizador.

A informação recebida na interface é devidamente tratada, sendo colocada nas devidas caixas de texto em frente a *Register Data* (Figura 4.5) e guardada no mesmo ficheiro onde já se encontram as informações previamente enviadas para o expansor.

```
int count = 0;
char inByte[] = " ";
void loop() {
 if(Serial.available()){
   char inByte = Serial.read();
   //Escrever para o porto A
    if(count < 8){
      if(inByte == '1'){
        expansor.digitalWrite(count + 1, HIGH);
      }
      else if(inByte == '0'){
        expansor.digitalWrite(count + 1, LOW);
      }
    }
    //Escrever para o porto B
    else if(count >= 8 and count < 16){
      if(inByte == '1'){
        expansor.digitalWrite(count + 1, HIGH);
      }
      else if(inByte == '0'){
        expansor.digitalWrite(count + 1, LOW);
      }
      //ler o que foi escrito em ambos os portos
      if(count == 15){
          for(int i = 1; i <=16; i++){
            Serial.print(expansor.digitalRead(i));
          }
      }
    }
   count = count + 1;
   delay(500);
  }
}
```

Figura 4.7 - Função loop no ambiente de desenvolvimento do Arduino.

4.2.4 Interface dos Conversores

A HV Remote tem três DACs (DAC7568) e um ADC (TLV2541). Cada um dos DACs tem 8 canais, o que perfaz 24 canais. As tensões de cada canal são enviadas para a malha de regulação associada a um PMT, sendo posteriormente amplificadas de forma a fornecer a alta tensão ao PMT. A HV Remote fornece a HV dedicada a 24 PMTs, tendo cada um deles a respetiva malha de regulação. Nesta malha de regulação existe ainda um divisor de tensão que permite que uma tensão de magnitude proporcional à alta tensão seja enviada para o multiplexador de 16-*bits* MCP506 e posteriormente lida pelo ADC, sendo de seguida recebida no computador.

Existem três abordagens a considerar na elaboração desta parte da interface de utilizador:

1. Teste individual do DAC;
2. Teste individual do ADC;
3. Teste de ambos, com ou sem ligação ao multiplexador.

Ambos os conversores serão testados com recurso a geradores de ruído uniforme. Os valores de teste encontram-se guardados em ficheiros, em formato analógico, que devem ser selecionados no início do envio de dados, através do devido painel da interface (Figura 4.8).

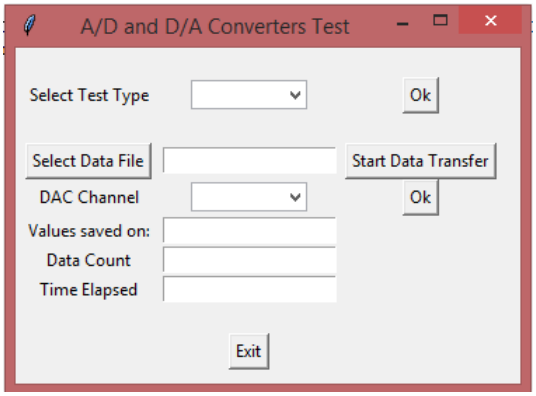


Figura 4.8 - Painel da interface em Python que permite efetuar o teste dos conversores.

O DAC7568 recebe uma palavra digital de 32-*bit* que permitem controlar a forma de distribuição de valores de tensão para os seus canais. É possível alterar a tensão de canais específicos ou de todos os canais simultaneamente. Dos 32-*bit* a enviar para o conversor apenas 12 correspondem à tensão que se pretende colocar num determinado canal, sendo os restantes *bits* de controlo. Através da interface, o utilizador pode selecionar qual o ficheiro de dados que quer utilizar para testar o conversor e qual o canal que quer testar. As tensões analógicas contidas no ficheiro são devidamente convertidas para formato digital e concatenadas na palavra de 32-*bit* apropriada para o canal escolhido.

Para comunicar através do computador com o DAC é necessário utilizar o expansor MCP23S17 como intermediário. É através deste que serão controlados dos pinos LDAC e SYNC do DAC. Estes pinos determinam se a operação de transmissão de dados é ou não bem sucedida (Figura 4.9). O pino LDAC deve, por norma, estar com o valor digital '1', sendo apenas colocado a '0' alguns microssegundos após o fim da conversão de dados. O pino SYNC determina o início da conversão dos dados recebidos no pino DIN. Apresenta o valor digital '1' sempre que não está a ocorrer uma conversão de dados, existindo uma transição para o valor '0' para dar início à conversão. Se este pino for colocado a '1' antes de terem sido convertidos os 32-*bit*, todo o processo é anulado.

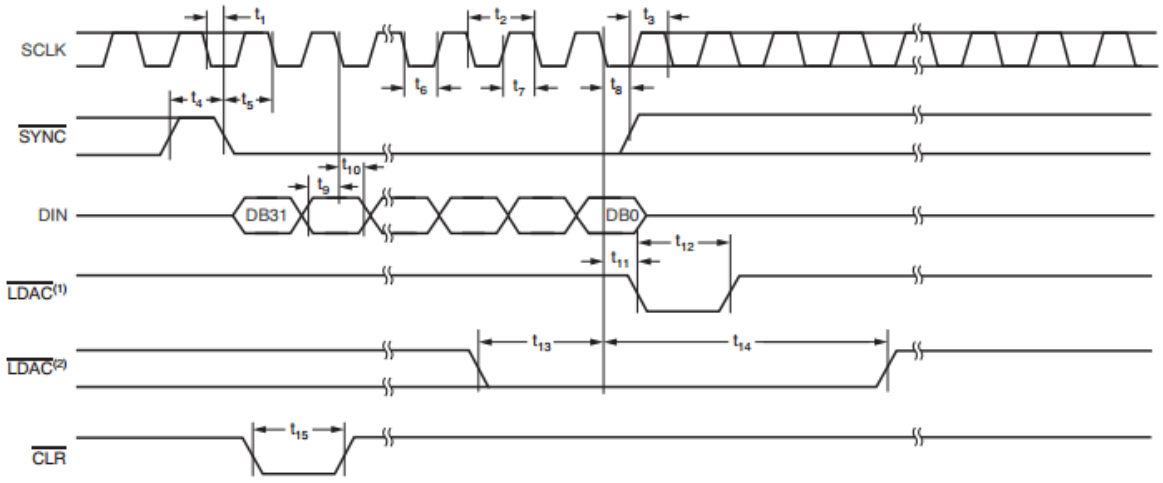


Figura 4.9 - Diagrama temporal dos valores de tensão a colocar nos pinos do DAC7568 para o envio de dados [14].

Como o valor dos pinos referidos anteriormente é independente do canal a ser testado, são definidos diretamente na interface do Arduino com recurso às funções descritas na seção anterior. A palavra de 32-*bit* enviada da interface em *Python* para a interface do Arduino é dividida em quatro palavras, de 8-*bit* cada uma, de forma a ser enviada através da biblioteca SPI do Arduino. Na função *setup* do Arduino são definidas as configurações do protocolo SPI, através de (4.8).

$$SPI.beginTransaction(SPISettings(50000000, MSBFIRST, SPI\_MODE0)) \quad (4.8)$$

A função *SPISettings* recebe três parâmetros:

1. A frequência do relógio máxima a que a transação deve ser efetuada;
2. O modo de envio de dados:
    a. *MSBFIRST*[32] permite o envio de dados no seu formato normal, com o *bit* mais significativo a ser enviado em primeiro.
    b. *LBSFIRST*[33] inverte o sentido da palavra binária, de forma a enviar o *bit* menos significativo em primeiro.
3. O modo de comunicação do protocolo SPI (descrição na primeira secção deste capítulo).

Na função *loop*, inicia-se a transação SPI, através de (4.9), e enviam-se os dados pretendidos, através de (4.10).

$$SPI.begin() \quad (4.9)$$

$$SPI.transfer(x) \quad (4.10)$$

O protocolo SPI tem uma particularidade no Arduino: para efetuar uma leitura de dados deve ser enviado um zero através da função (4.10). Para terminar a transação deve-se utilizar a expressão (4.11).

$$SPI.endTransaction() \quad (4.11)$$

Os valores recebidos são enviados de volta para a interface em *Python*, onde são devidamente tratados e guardados num ficheiro. Por norma, o protocolo SPI envolve o controlo do pino CS (*chip select*[34]) do dispositivo para o qual escreve ou do qual lê valores. Neste caso específico, o pino CS do DAC corresponde ao pino SYNC que, como foi descrito anteriormente, é controlado com recurso ao expansor MCP23S17.

No que toca à interface de utilizador em *Python*, o procedimento para testar individualmente o ADC é muito semelhante ao descrito anteriormente. Não existe necessidade de converter os dados de teste porque as tensões de entrada do ADC também são analógicas. No entanto, enquanto a receção de dados do ADC utiliza o protocolo SPI, os dados analógicos a testar não são enviados por SPI para o conversor. Existe uma ligação direta entre o Arduino e o ADC.

O Arduino contém uma função, (4.12), que permite a utilização de alguns dos seus pinos para escrever valores analógicos [47], sendo *x* o valor que se pretende enviar através do Arduino. No caso do Arduino Uno, esta funcionalidade está disponível nos pinos 3, 5, 6, 9, 10 e 11.

$$analogWrite(x) \quad (4.12)$$

Esta função utiliza uma técnica denominada por *Pulse Width Modulation*, PWM [48]. É criada uma onda quadrada que oscila entre os valores digitais '1' e '0' e permite simular valores se tensão entre 0 e 5 V, através da variação do tempo em que a onda apresenta cada um dos valores digitais, Figura 4.10. A frequência da onda gerada através dos pinos 5 e 6 é de 1 kHz, enquanto dos outros pinos é cerca de 500 Hz [49].

[32] Sigla de *Most Significant Bit First*.
[33] Sigla de *Least Significant Bit First.*
[34] Este pino permite selecionar um dispositivo como *slave* durante a utilização do protocolo SPI (descrição efetuada na primeira seção deste capítulo).

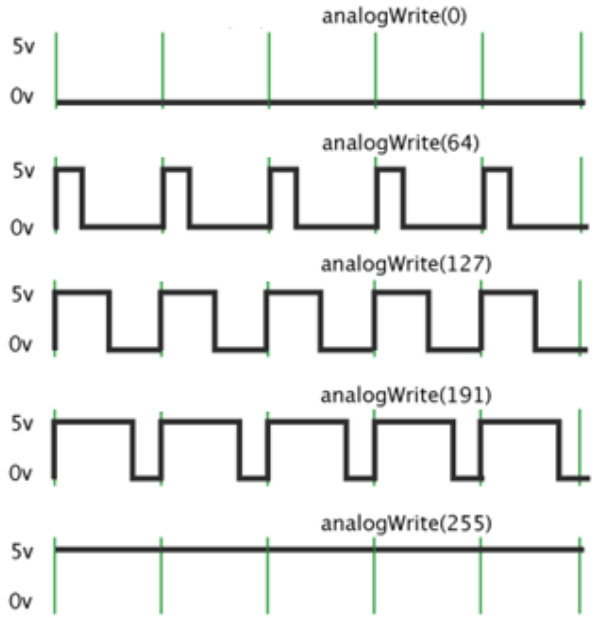


Figura 4.10 - Exemplo de valores utilizados com a função analogWrite e ondas quadradas geradas (PWM).

O valor enviado pela função (4.12) não é o valor de tensão real que se quer enviar para o ADC. Esta função recebe um valor codificado entre 0, que corresponde a 0 V, e 255, que corresponde a 5 V. Com recurso a uma simples regra de três simples é possível chegar à expressão (4.13), que permite a obtenção do valor codificado a enviar através de (4.12).

$$Valor\ codificado = 51 \times Valor\ Analógico \tag{4.13}$$

Entre o envio de valores de tensão para o ADC e a leitura das suas conversões, deve existir uma pausa programada de acordo com os valores indicados na ficha técnica do componente. Para ler os valores convertidos pelo ADC, utiliza-se o protocolo SPI. O primeiro passo é colocar o CS do ADC a '0' através de (4.14).

$$digitalWrite(pino\ CS, LOW) \tag{4.14}$$

Faz-se então a chamada à função (4.10) de forma a receber os valores convertidos pelo ADC. Como o conversor é de 12-*bit*, o valor recebido estará codificado entre 0 e 4095 ($2^{12} = 4096$), ou seja, o ADC apresenta 4096 valores digitais de saída possíveis. Desta forma, recorre-se novamente a uma regra de três simples para determinar o valor real da tensão recebida através de (4.15).

$$Valor\ Analógico = \frac{5 \times Valor\ Codificado}{4096} \tag{4.15}$$

A função (4.10) apenas permite a transferência de 8-*bit* por chamada. Como tal, é necessário efetuar duas chamadas à função. Além disso, antes de recorrer a (4.15) para converter o valor codificado para um valor de tensão analógico, este valor deve ser devidamente processado. A segunda chamada à função (4.10) devolve 8-*bit* dos quais apenas queremos 4, de forma a perfazer os 12-*bit* convertidos pelo ADC. As funções observadas em (4.16), (4.17) e (4.18) permitem efetuar o devido deslocamento de *bits* de forma a obter a palavra digital de saída do ADC correspondente ao valor de entrada enviado.

$$byte1 = (byte1\ \&\ 127) \tag{4.16}$$

$$byte2 = byte2 >> 3 \tag{4.17}$$

$$bitnum = (word(byte1) << 5) \mid byte2 \quad (4.18)$$

Após as operações descritas anteriormente, desselecciona-se o ADC através do comando em (4.19).

$$digitalWrite(pino\ CS, HIGH) \quad (4.19)$$

Os valores são enviados para a interface em *Python* onde, tal como os valores recebidos do DAC, são guardados num ficheiro.

Por último existe ainda a possibilidade de testar o DAC e o ADC simultaneamente. Existem algumas diferenças a nível de programação para efetuar estes testes, mas as principais diferenças são ao nível de montagem, sendo necessário recorrer a duas montagens diferentes:

1. Os valores serão enviados através do Arduino para o expansor MCP23S17 que, por sua vez, enviará os valores de configuração necessários para o DAC7568, para o multiplexador MPC506 e para o ADC TLV2545.
2. Os valores serão enviados através do Arduino para o expansor MCP23S17 que, por sua vez, enviará os valores de configuração necessários para o DAC7568 e para o ADC TLV2545.

A nível de interface, tudo o que já foi descrito neste capítulo é reutilizado. Entrar-se-á em detalhes sobre a configuração dos parâmetros dos componentes referidos no próximo capítulo.

4.2.5 Interface para testes de desempenho da placa de controlo

Definem-se como testes de desempenho da placa de controlo a leitura de todas as tensões fixas e da tensão proporcional à temperatura que se encontram na carta. Estas medições são efetuadas com recurso a um osciloscópio da *National Instruments* (Tektronix TDS2014B) e à biblioteca *pyvisa* do Arduino.

A biblioteca *pyvisa* permite detetar recursos, através da expressão (4.20), da *National Instruments* que sejam conectados ao computador através da porta USB. É possível obter uma lista desses recursos, através de (4.21) e selecionar aquele que pretendemos utilizar.

$$rm = visa.ResourseManager() \quad (4.20)$$

$$rm\_list = rm.list\_resources() \quad (4.21)$$

As medidas são sempre efetuadas com recurso à mesma ponta de prova e ao mesmo canal do osciloscópio, sendo necessário colocar a ponta de prova no pino de teste cujo valor analógico se pretende obter. Recorrendo a esta opção da interface de utilizador, Figura 4.11, é possível obter os seguintes valores:

1. Tensão de referência associado ao componente AD589;
2. Temperatura, em *volt*, associada ao componente TMP17, procedendo-se posteriormente à conversão para ˚C através de (2.1);
3. Tensão de referência do ADC TLV2541;
4. Tensões de referência dos DACs DAC7568 e MAX5725;
5. Frequência do sinal de relógio comum a todos os componentes da placa.

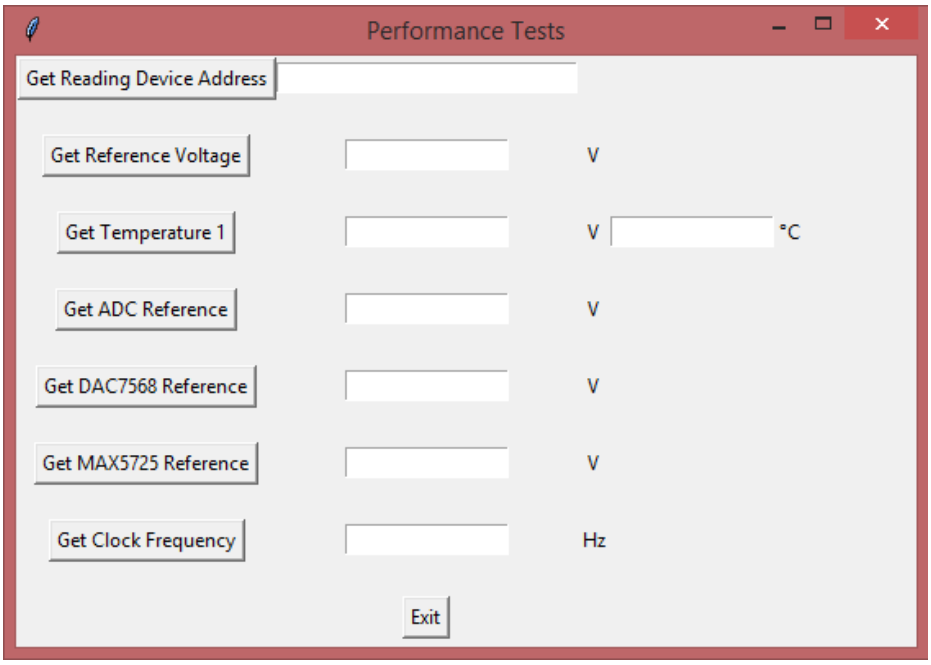


Figura 4.11 - Painel da interface em Python que permite efetuar o teste de desempenho.

5 TESTES EFETUADOS E ANÁLISE DOS RESPETIVOS RESULTADOS

O plano de testes associado à primeira versão da placa de controlo envolvia cinco tipos de teste diferentes, já mencionados no capítulo anterior:

1. Teste ao expansor série/paralelo MCP23S17;
2. Teste ao DAC DAC7568 de 12-*bit*;
3. Teste ao ADC TLV2541 de 12-*bit*;
4. Teste a ambos os conversores, DAC7568 e TLV2541;
5. Teste a ambos os conversores em conjunto com o multiplexador de 16-*bit* MPC506.

Com recurso à primeira versão da placa de controlo foi possível efetuar os testes ao expansor MCP23S17. Durante a implementação dos testes a aplicar ao DAC e ao ADC foi verificada a impossibilidade de comunicação com o mesmo. Após algumas tentativas, foram descobertos erros no desenho da placa que podem explicar o seu mau funcionamento, bem como a necessidade de adição de pontos de teste.

Entre os erros descobertos, é de salientar a má ligação do DAC7568 aos planos de massa da placa: uma das suas massas analógicas encontrava-se ligada ao plano de massa digital. Além disso, devido à necessidade de sincronização entre as linhas de dados do protocolo SPI, era necessária a observação dos sinais associados aos pinos SYNC e LDAC. Outra fonte de erro de importância foi a ligação entre os planos de massa digital e analógica não ser convenientemente efetuada até uma fase já avançada dos testes. Aparentemente, nenhum componente sofreu qualquer tipo de dano devido a esta má ligação, mas não é possível garantir que tal não aconteceu.

Considerando os factos descritos no parágrafo anterior e ainda o aquecimento verificado na tensão de referência REF02, optou-se pela elaboração de uma nova versão da placa de controlo, já descrita no Capítulo 2. Esta placa já se encontra em fase de assemblagem mas optou-se por não incluir os seus testes nesta tese de mestrado, devido ao alargado período temporal associado a estes. O esquema de testes inicialmente definido foi por isso alterado.

Os testes ao expansor MCP23S17 já tinham sido efetuados e são incluídos neste capítulo. O ADC TLV2541 não é disponibilizado em pacotes que permitam o seu teste em placas de teste individuais, pelo que foi pedida uma amostra do ADC MAX189 de 12-*bit* e com opção de comunicação por SPI, uma opção muito semelhante ao ADC TLV251. Devido à demora em encontrar uma alternativa para testar um DAC, optou-se por apenas realizar testes ao ADC. Para além do MAX189, foi ainda testado o ADC do Arduino Due.

Apesar de o plano inicial envolver testes com um gerador de ruído Gaussiano, idealmente são necessários 2^{25} valores de teste [50]. Considerando ainda que a montagem ideal de teste seria com o 2º protótipo da placa de controlo, o teste com valores gerados pelo gerador Gaussiano iria aumentar o tempo de realização de testes, não estando de acordo com o tempo expectável de realização desta dissertação. No entanto, estes testes serão realizados em trabalho futuro.

Descrevem-se de seguida os testes efetuados ao expansor MCP23S17, ao ADC MAX189 e ao ADC do Arduino Due. Para os primeiros dois testes foi utilizado o Arduino Uno.

5.1 TESTE DO EXPANSOR SÉRIE/PARALELO

O teste do expansor MCP23S17 foi efetuado com recurso à interface de utilizador desenvolvida em *Python*, descrita no capítulo anterior. Este teste consistiu no envio de palavras de 16-*bit* para o

expansor, através do protocolo SPI, e na verificação dos dados recebidos. Um esquema montagem experimental utilizada encontra-se representada na Figura 5.1, e uma fotografia da mesma encontra-se na Figura 5.2.

Foram enviadas vinte palavras diferentes para o expansor e as mesmas vinte palavras foram recebidas de volta. Este teste, apesar de bastante simples, permite garantir a fiabilidade do expansor, uma vez que este irá distribuir os valores de configuração necessários ao funcionamento dos restantes componentes tanto da placa de controlo como da HV Remote.

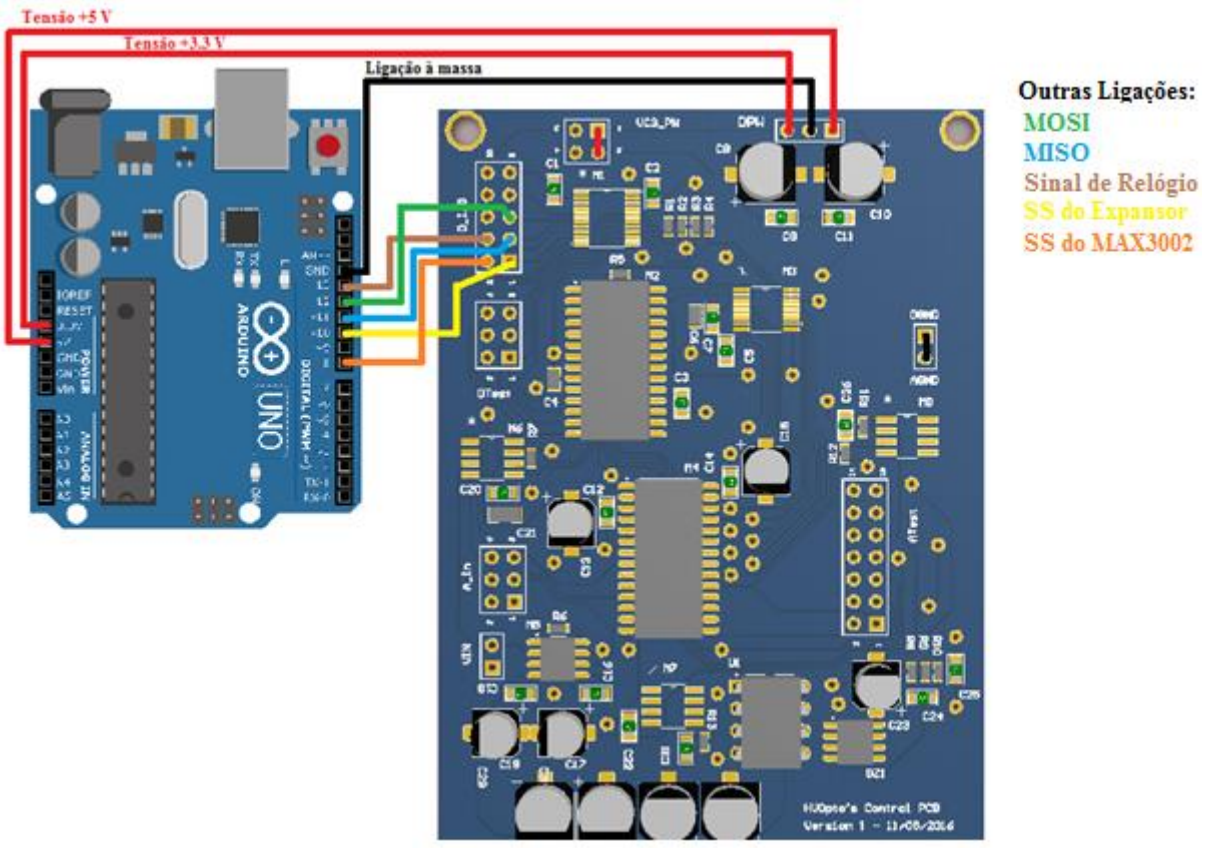


Figura 5.1 - Montagem experimental associada ao teste do expansor MCP23S17.

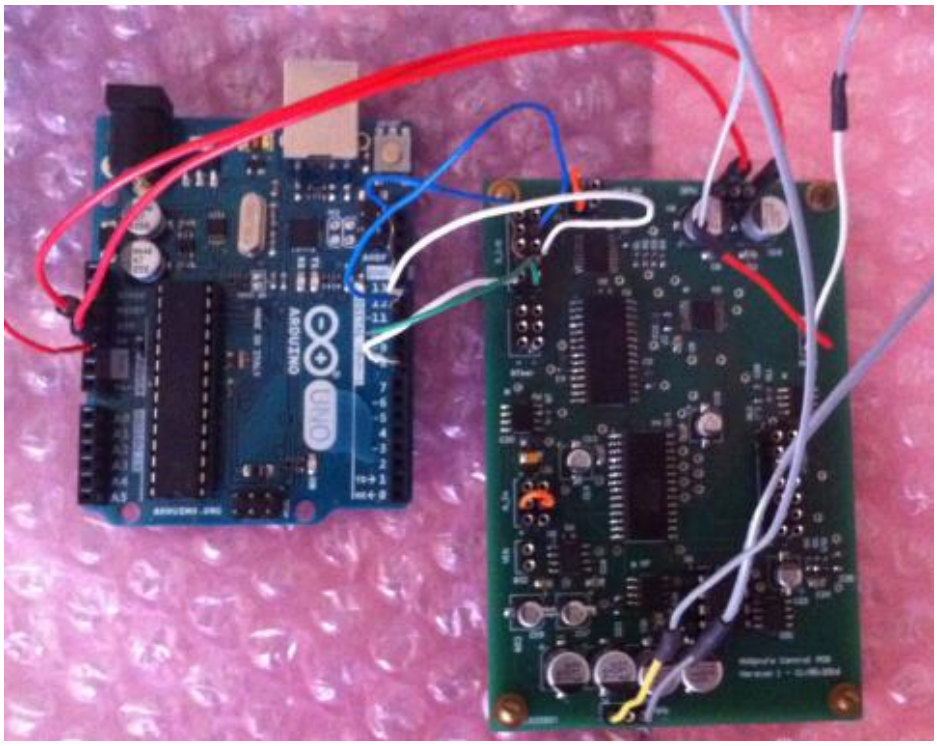


Figura 5.2 - Fotografia da montagem experimental utilizada, com um Arduino Uno e a primeira versão da placa de controlo.

5.2 TESTE DO ADC MAX189

O ADC MAX189 é um conversor de 12-*bit* compatível com o protocolo SPI, opera entre 0 e 5 V e recebe como tensão de alimentação 5 V [51]. Este conversor foi escolhido pelo facto de os testes a elaborar poderem ser transpostos para o TLV2541, sem efetuar quaisquer tipo de alterações ao código já preparado tanto na interface de utilizador em *Python* como no Arduino. Além disso, apresenta o encapsulamento PDIP[35] que permite a sua utilização em placas de teste. A comunicação com o ADC é efetuada através de ligação direta aos pinos do Arduino Uno associados ao protocolo SPI, a respetiva montagem experimental encontra-se representada nas Figuras 5.3 e 5.4.

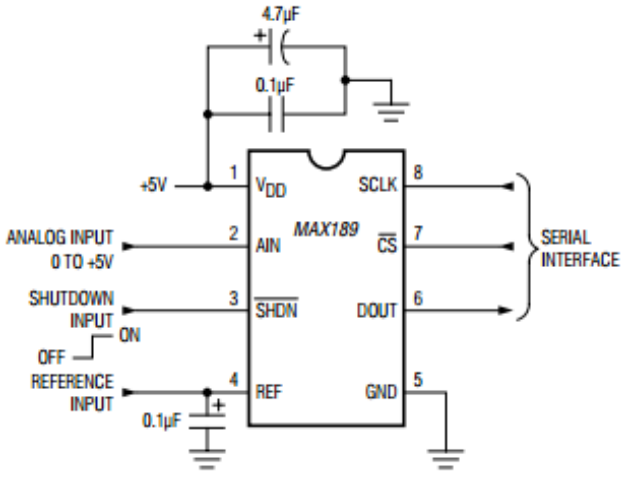


Figura 5.3 – Diagrama operacional do ADC MAX189 [51].

[35] Sigla de *Package Dual In-Line Pin*.

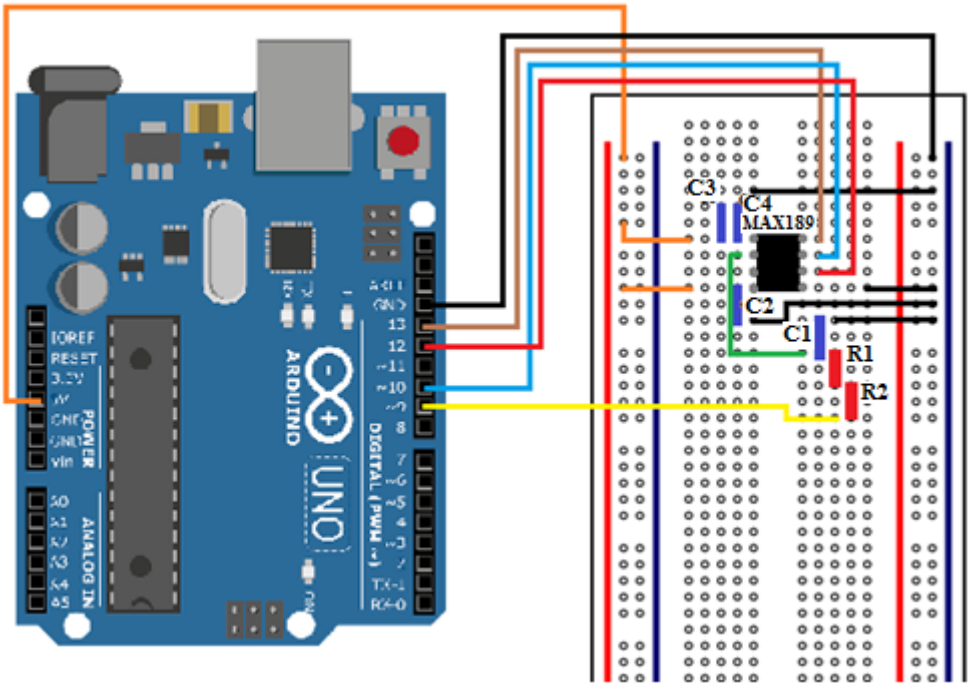


Figura 5.4 - Diagrama da montagem experimental utilizada para testar o ADC MAX189.

O Arduino Uno dispõe de quatro pinos configurados por defeito para utilização do protocolo SPI:

1. O pino 10 corresponde ao *Chip Select* que permite selecionar um dispositivo;
2. O pino 11 corresponde ao *MOSI* que permite enviar dados para um dispositivo;
3. O pino 12 corresponde ao *MISO* que permite receber dados de um dispositivo;
4. O pino 13 corresponde ao *CLOCK*, de forma a assegura sincronização entre os sinais de relógio do dispositivo mestre e dos dispositivos com os quais se pretendem efetuar a comunicação.

Especificamente para o teste isolado de um ADC não é utilizado o pino *MOSI*, tal como foi referido no capítulo anterior, pois os dados são enviados através de um pino PWM do Arduino, mais especificamente, o pino 9.

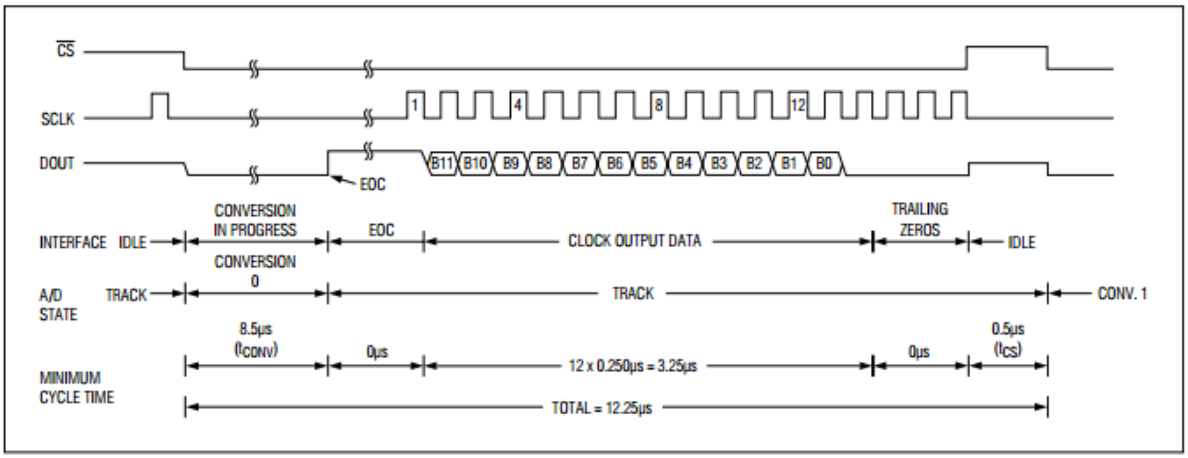


Figura 5.5 - Diagrama temporal de comunicação com o ADC MAX189 [51].

A fase inicial dos testes efetuados consistiu na sincronização temporal dos parâmetros enviados para o ADC através da interface SPI, Figura 5.5. Nomeadamente, a obrigatoriedade de manter o *chip select* a

‘0’, desde a inserção de um dado no pino de entrado do ADC até à sua conversão. O sinal de relógio deste protocolo só é ativado quando está a acontecer uma conversão de dados, sendo por isso um processo automático. Apenas foi necessário adicionar um atraso de 8,5 µs na comunicação para garantir que o processo decorreria sem qualquer problema.

A segunda fase dos testes prendeu-se com a verificação do sinal emitido pela PWM gerada pelo Arduino. Ao efetuar testes iniciais, de forma a garantir a integridade dos dados durante a sua transmissão, verificou-se que a onda gerada através de PWM apresentava uma grande variação de amplitude face às tensões analógicas pretendidas. Esta variação está associada à frequência de onda pré-definida (490.20 Hz, [49]) para o pino utilizado (neste caso, o pino 9). Para tornar o sinal o mais aproximado possível de um sinal puramente analógico foi necessário aumentar a frequência do sinal, com recurso ao comando [49]:

$$\mathrm{TCCR1B} = \mathrm{TCCR1B} \,\&\, \mathrm{B11111000} \mid \mathrm{B00000010} \tag{5.1}$$

Do comando anterior, a única parte personalizável é a palavra binária a seguir ao operador OR (representado pelo símbolo |), sendo essa palavra que define a nova frequência, cerca de 4 kHz, associada ao sinal emitido através do pino 9. Além disso, foi adicionado um filtro passa-baixo (condensador C1 e resistências R1 e R2 que podem ser vistas na Figura 5.4) para assegurar, em conjunto com a alteração do sistema, um sinal de entrada analógico estável para o ADC. Com estas alterações, o envio de dados através de PWM tornou-se viável, não tendo sido verificadas variações significantes nos valores recebidos do ADC, face aos enviados.

Foram enviados 2^{23} valores de teste para o ADC, para o teste de um dos geradores uniformes, tendo esses valores sido recebidos e guardados em ficheiro, para posterior tratamento através do software *Matlab*. Estes valores de teste foram gerados com recurso a geradores de números aleatórios implementados em *Matlab*, nomeadamente, o gerador Mersenne Twister e o gerador Congruente Multiplicativo. Ambos os geradores são uniformes.

Os valores gerados através do código em *Matlab* (Anexos 8.15 e 8.16) foram utilizados como ficheiro de entrada na interface de utilizador em *Python* e enviados para o ADC. Para garantir a sincronização dos dados enviados para o Arduino com os dados recebidos na interface, verificou-se ser necessário introduzir um atraso de cerca de 1,5 s entre o envio e a receção dos dados. Assim, o processo tornou-se demasiado lento, demorando cerca de uma semana a enviar todos os valores de teste. Para ser possível proceder com os testes planeados, foi elaborada uma alteração no envio dos dados para o Arduino. Em vez da interface de utilizador, foi utilizado um código (Anexo 8.12) no ambiente de desenvolvimento *Processing* para ler os valores de dados gerados pelo *Matlab*, enviá-los para o Arduino, receber os valores provenientes do ADC e guardá-los num ficheiro. Este novo código possibilita as mesmas funcionalidades que a interface em Python, à exceção da contagem do tempo decorrido.

Com a alteração referida no parágrafo anterior, a transferência de dados para um teste do ADC, com o número de amostras referido, passou a demorar apenas três dias.

5.2.1 Gerador Uniforme - Mersenne Twister

O ficheiro de texto, com os valores recebidos do ADC, foi lido diretamente para um vetor e foram efetuadas algumas verificações:

1. Se alguma posição do vetor não apresentasse um valor numérico válido, essa posição seria excluída pois poderia resultar de erros de leitura;

2. O ADC em teste apresenta um valor de tensão de entrada entre 0 e 5 V e os valores gerados em *Matlab* também se encontram nesse intervalo, se existissem valores superiores seriam considerados erros de fim de escala e excluídos.

O primeiro objetivo da análise de dados passou por gerar o histograma associado aos dados obtidos. Os valores de tensões do ficheiro obtido encontram-se entre 0 e 5 V, enquanto o histograma é gerado de acordo com o número de *bits* do conversor. Um ADC de 12-*bit* implica um histograma com 4096 códigos possíveis. Como tal, é necessário converter as tensões que se encontram no ficheiro em códigos que possam ser associados ao histograma, através de (5.2).

$$Noise(x) = Noise(x) * \frac{4095}{5} \tag{5.2}$$

Sendo $Noise(x)$ o vetor onde são guardados os valores de tensão, o valor 4095 representa o valor de códigos possíveis $(2^N - 1)$ e o valor 5 a tensão de referência do ADC.

Com os valores de tensão convertidos para os respetivos códigos, foi possível obter o histograma associado a estes valores de tensão, que pode ser observado na Figura 5.6. É importante observar que este histograma é bastante linear, ou seja, bastante uniforme.

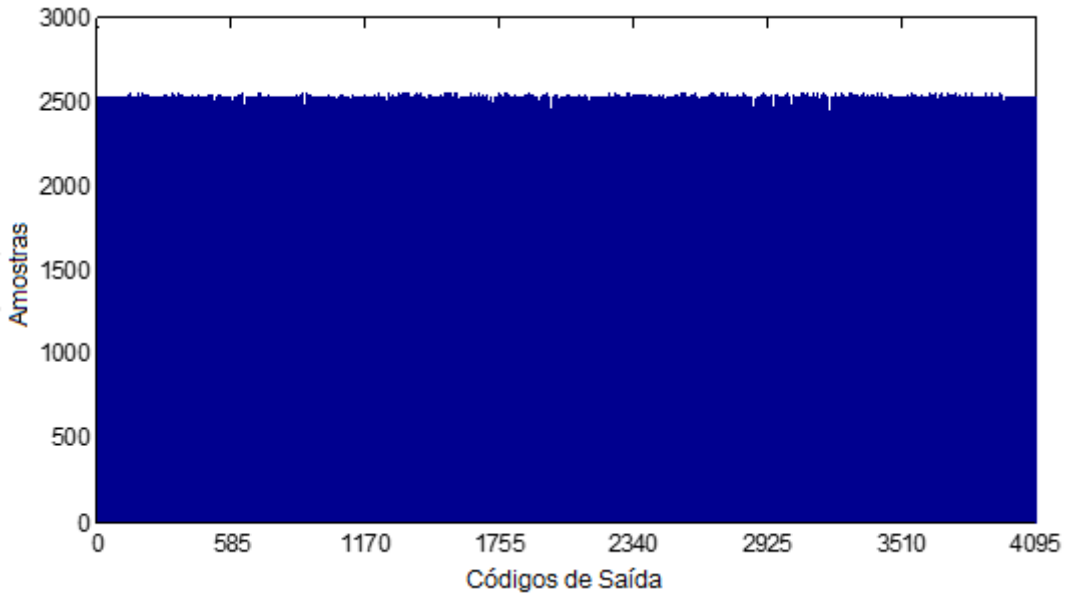


Figura 5.6 - Histograma associado aos valores testados com o gerador uniforme Mersenne Twister.

Como foi referido no Capítulo 3, é importante proceder à correção do erro de ganho e da tensão de desvio antes de calcular os parâmetros de não-linearidade do ADC. É importante referir que os histogramas uniformes obtidos pelo método utilizado nesta dissertação apresentam duas grandes vantagens face a histogramas tradicionais:

1. A não consideração dos códigos de saída nos extremos do histograma significa que o mesmo apenas é caracterizado por um único valor, $Hist_{ideal}$.
2. O cálculo dos parâmetros do conversor fica simplificado.

A tensão de desvio é o valor médio da tensão de entrada quando nesta se deveria verificar uma tensão nula, ou seja, quando o código na saída do ADC é um '0'. Considerando o referido no parágrafo anterior, para histogramas uniformes, o cálculo da tensão de desvio é efetuado com recurso a uma estimativa, representada na expressão (5.3).

$$V_{OS} = \frac{Hist(2^N) - Hist(1)}{2 \times Hist_{ideal}} \quad (5.3)$$

Sendo $Hist(2^N) - Hist(1)$ a diferença entre os números de amostras obtidos experimentalmente para os códigos de saída extremos e $Hist_{ideal} = \frac{\mathrm{NA}}{2^{\mathrm{N}}}$, sendo NA o número de amostras consideradas e N a resolução do conversor.

Para além da tensão de desvio, é necessário proceder ao cálculo do erro de ganho. Este erro é definido como a diferença entre a curva de transferência ideal e a real, após a correção da tensão de desvio, e é dado pela expressão (5.4).

$$Erro\ de\ Ganho = \frac{1}{(2^N - 2) \times Hist_{ideal}} \sum_{i=2}^{2^N - 1} Hist(i) \quad (5.4)$$

Devido à dispersão no número de amostras obtido experimentalmente para cada código, o erro de ganho deve ser calculado para um elevado número de níveis de quantificação ou mesmo para todos eles, excluindo os valores extremos, e posteriormente efetuando a média do resultado obtido.

Após estes cálculos, procede-se para o cálculo das não-linearidades diferencial e integral. A DNL pode ser calculada através de (5.5) e a INL através de (5.6).

$$DNL(i) = \frac{\left(\frac{Hist(i)}{Erro\ de\ Ganho}\right) - Hist_{ideal}}{Hist_{ideal}} \quad (5.5)$$

Sendo a primeira parcela da subtração a correção do erro de ganho aplicada à não-linearidade diferencial. A não-linearidade integral, tal como referido no Capítulo 3, corresponde à soma da DNL para todos os níveis de quantificação.

$$INL(i) = \sum_{j=1}^{i} DNL(j) \quad (5.6)$$

Idealmente, o valor da DNL deveria ser nulo, pelo que quando mais perto de 0 se encontrar, melhor é a qualidade do ADC em teste, sendo entre ± 0.5 LSB o valor recomendado. Já o valor da INL deve encontrar-se entre ± 1 LSB. Para o teste deste ADC, os resultados obtidos experimentalmente, bem como os valores fornecidos pelo fabricante, podem ser vistos na Tabela 3.

Parâmetro	**Valores Experimentais (LSB)**	**Valores do Fabricante (LSB)**
Tensão de Desvio, V_{OS}	- 0.43	± 1.50
Erro de Ganho	0.97	± 1
DNL Máxima	0.04	1
DNL Mínima	-0.08	- 1
INL Máxima	1.15	-
INL Mínima	-1.20	-

Tabela 3 - Valores obtidos experimentalmente e valores fornecidos pelo fabricante [51] para os parâmetros do ADC MAX189.

O valor negativo da tensão de desvio significa o primeiro nível de quantificação do ADC, em vez de corresponder a 0 LSB, correspondia ao LSB – 0.43. O valor do erro de ganho significa que, para o último nível de quantificação, em vez de se obter o valor de tensão correspondente ao código 4096, obteve-se praticamente mais 1 LSB. O intervalo de valores de DNL obtido é muito satisfatório, uma vez que se encontra dentro do intervalo esperado. Já o intervalo de valores da INL ultrapassa um pouco o valor

pretendido, entre ± 1 LSB. Esta diferença face aos valores esperados pode ter origem no número de amostras utilizado (um maior número de amostra implica uma maior uniformidade) ou no facto de o sinal utilizado como sinal de entrada do ADC não ser um sinal analógico puro, mas sim um sinal PWM.

Apesar dos valores superiores da INL, os restantes valores estão de acordo com os valores fornecidos pelo fabricante [51]. Os valores obtidos para a DNL e INL podem ser observados na Figura 5.7.

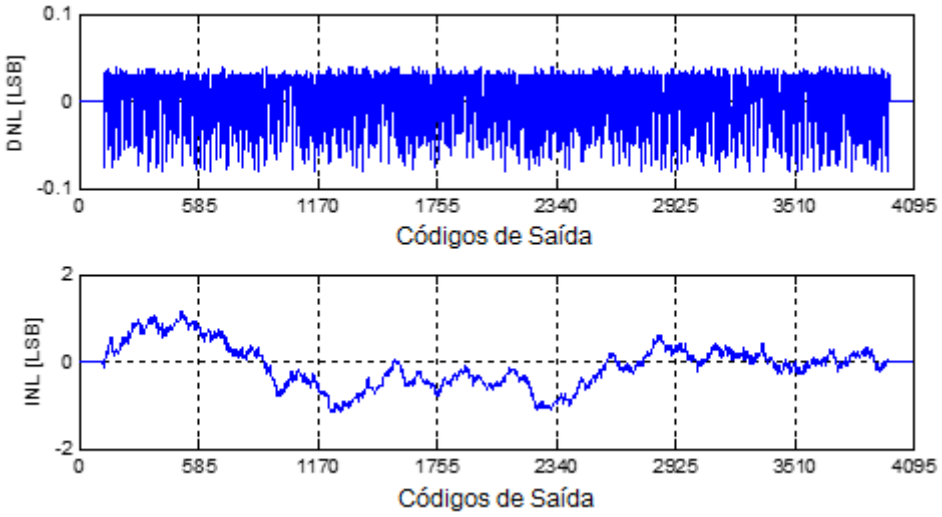


Figura 5.7 - Gráficos dos valores de DNL e INL obtidos para os valores do gerador uniforme Mersenne Twister.

5.3 TESTE DO ADC DO ARDUINO DUE

O Arduino Due possui uma grande vantagem face ao Arduino Uno: tem incluídos dois DACs. Ao contrário do Arduino Uno, com o Due é possível gerar um sinal analógico puro, sendo este mais fiável do que o sinal analógico gerado por PWM utilizado para testar o ADC MAX189. A única contrapartida na sua utilização é o facto de a tensão de entrada máxima dos seus pinos analógicos ser 3.3 V. Uma tensão superior pode danificar os pinos do Arduino. Além disso, ambos os DACs tem uma gama limitada: entre 0.55 V e 2.75 V.

Para efetuar este teste, foi utilizado o gerador uniforme Congruente Multiplicativo. Os valores de tensão de entrada do ADC foram gerados, mais uma vez, a partir do código em *Matlab*, considerando o gerador em questão. A montagem experimental efetuada, Figura 5.8, é muito mais simples que as anteriores. Como o objetivo é o teste do ADC embutido no Arduino Due, através da geração de valores de entrada analógicos puros com recurso ao DAC do Arduino Due, a única ligação a efetuar é entre um dos seus DACs (DAC0 e DAC1) e uma das suas entradas analógicas (A0 a A11), Figura 5.8.

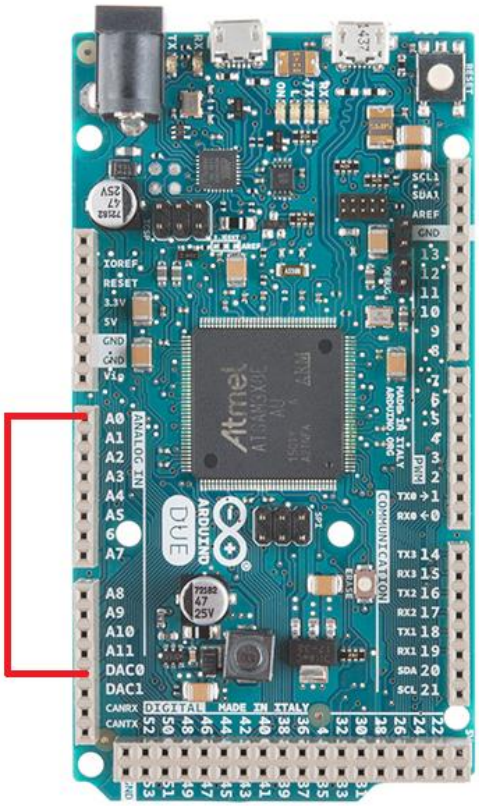


Figura 5.8 - Diagrama da montagem experimental utilizada para testar o ADC do Arduino Due.

Devido à limitação na gama de valores, tanto das entradas analógicas como das saídas analógica, é necessário converter os valores de tensão gerados com o *Matlab* para a devida escala. No entanto, como a tensão máxima de saída do DAC é inferior à tensão máxima de entrada do ADC, é possível fazer-se uma simples correspondência entre os valores enviados para o Arduino, pelo DAC, e os recebidos pelo ADC. Para tal, começou-se por enviar todos os códigos entre 0 e 4096 para o DAC e registar os valores recebidos através do ADC. Este procedimento foi repetido cinco vezes, efetuando-se a média dos valores recebidos e posteriormente foi feita a relação entre os valores enviados e os recebidos.

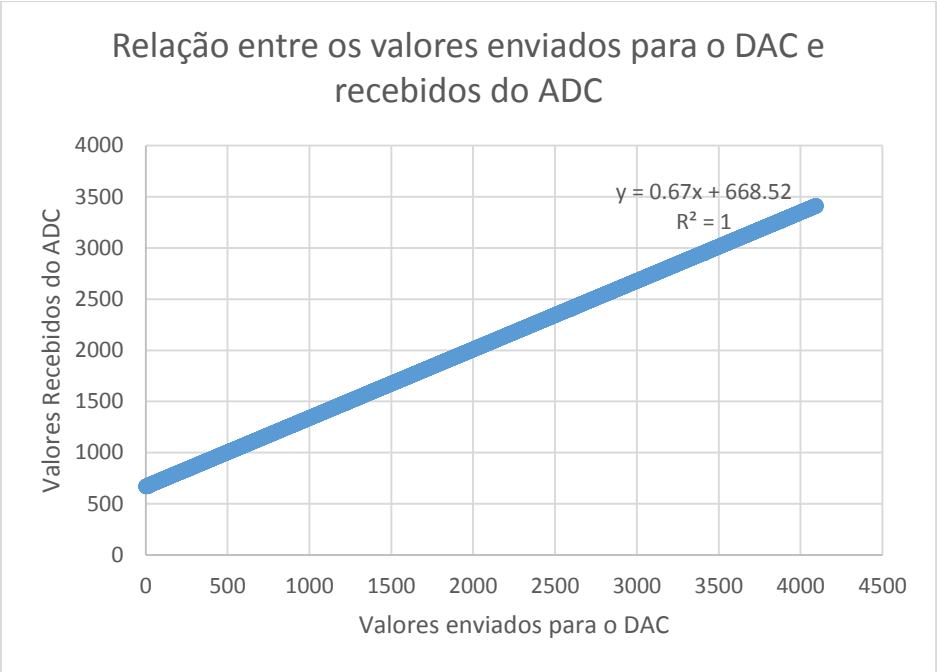


Figura 5.9 - Gráfico da relação entre os valores enviados para o DAC e os valores recebidos do ADC.

É importante referir que, mais uma vez, os valores a enviar para o Arduino foram convertidos de acordo com a expressão (4.13). Efetuou-se a regressão linear da reta obtida no gráfico da Figura 5.9, obtendo-se assim a relação entre os valores enviados para o DAC e os recebidos do ADC.

$$Valores\ Recebidos\ do\ ADC = 0.67 \times Valores\ Enviados\ para\ o\ DAC + 668.52 \qquad (5.7)$$

Após a obtenção da relação entre ambos os valores, foi possível enviar os valores de tensão gerados em *Matlab* para o Arduino. Os dados recebidos do Arduino foram, mais uma vez, armazenado num ficheiro, tendo o seu tratamento sido efetuado de acordo com o descrito em 5.2.1. O histograma associado aos dados pode ser observado em Figura 5.10. Apesar deste histograma também se apresentar muito uniforme, são observados mais picos que no histograma associado ao gerador uniforme Mersenne Twister, o que se pode dever a erros do DAC para além dos do ADC em teste. Apesar do histograma obtido, os valores da DNL e da INL encontram-se dentro dos intervalos expectáveis.

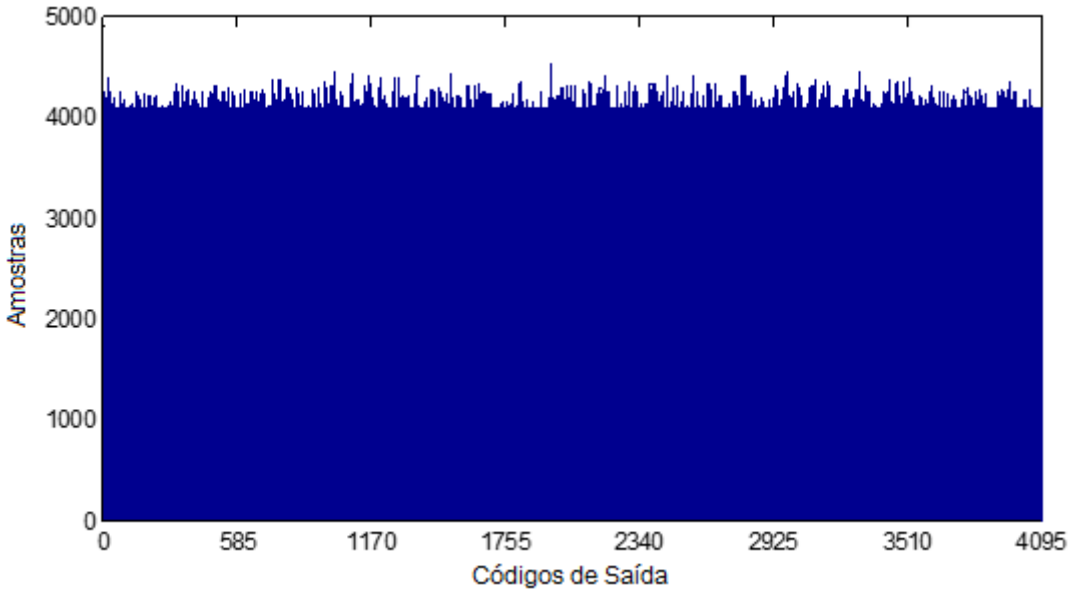


Figura 5.10 - Histograma associado aos valores testados com o gerador uniforme Congruente Multiplicativo.

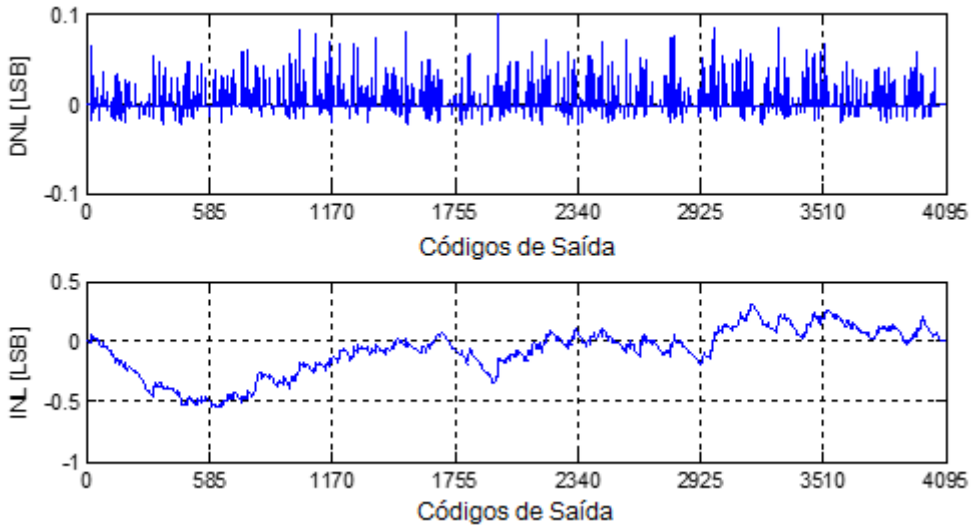


Figura 5.11 - Gráficos dos valores de DNL e INL obtidos para os valores gerados através do gerador uniforme Multiplicativo Congruente.

Relativamente à Figura 5.11 e comparativamente a Figura 5.7, observa-se de imediato que os valores da INL para este gerador são muito inferiores aos valores obtidos para o gerador Mersenne Twister. Estes valores confirmam a suspeita antes mencionada: os valores da INL apresentam-se menores para sinais de entrada analógicos puros.

Apesar da melhoria nos valores da INL, verificou-se que todos os outros valores apresentam piores resultados que os do gerador Mersenne Twister. A tensão de desvio e o erro de ganho, apesar de serem muito inferiores que os valores típicos indicados pelo fabricante [52], são superiores aos obtidos na secção anterior. O intervalo de valores da DNL também é muito superior ao obtido anteriormente. Apesar dos valores para a DNL e a INL poderem ser considerados muito bons, comparativamente ao geral de valores obtidos para o gerador Mersenne Twister, devido aos picos observados no histograma, Figura 5.10, são valores menos satisfatórios. Os valores obtidos para este teste podem ser observados na Tabela 4.

Parâmetro	Valores Experimentais (LSB)	Valores do Fabricante (LSB)
Tensão de Desvio, V_{OS}	- 0.76	11.50
Erro de Ganho	1.52	- 23
DNL Máxima	0.10	0.50
DNL Mínima	- 0.02	- 0.50
INL Máxima	- 0.56	1
INL Mínima	0.32	- 1

Tabela 4 - Valores obtidos experimentalmente e valores fornecidos pelo fabricante [52] para os parâmetros do ADC do Arduino Due.

6 CONCLUSÕES E TRABALHO FUTURO

O teste dos componentes que compõem a carta HV Remote é fundamental para garantir o seu bom funcionamento, antes da introdução da alta tensão. O objetivo desta dissertação foi o desenvolvimento de uma placa de controlo, em altas tensões, que permitisse o teste de todos os componentes utilizados na HV Remote, bem como o desenvolvimento de uma interface de utilizador que permitisse a comunicação com a placa de controlo.

Uma primeira ronda de testes revelou alguns problemas na conceção do 1º protótipo da placa de controlo. A existência de uma ligação errada à massa no DAC7568, a inexistência de pontos de teste em todos os sinais e o aquecimento de um dos seus componentes revelaram a necessidade de desenvolvimento de um 2º protótipo da placa de controlo. Neste 2º protótipo, os problemas enumerados anteriormente foram endereçados e foi adicionado um novo DAC, MAX5725, como alternativa ao DAC7568, devido a problemas de comunicação com este. Este protótipo foi desenvolvido durante o mês de Maio, já tendo sido recebido mas a assemblagem dos seus componentes ainda não foi efetuada. Por esse motivo, os testes a este protótipo só serão realizados futuramente.

A interface de utilizador desenvolvida encontra-se completamente funcional e pronta para efetuar testes ao 2º protótipo da placa de controlo, sendo apenas necessário efetuar alterações para proceder à comunicação com o DAC MAX5725. A interface permite efetuar testes ao expansor série/paralelo MCP23S17, aos conversores ADC e DAC e testes de desempenho à carta de controlo. Foram efetuados testes ao expansor série/paralelo, ao ADC MAX189 e ao ADC do Arduino Due.

Os testes efetuados ao expansor série/paralelo permitem concluir que este expansor recebe, sem quaisquer falhas, as definições definidas pelo utilizador, sendo isso um fator fundamental pois na HV Remote existem três expansores que vão permitir a configuração de todos os seus componentes.

Para os testes do ADC MAX189 e do ADC do Arduino Due foram gerados valores de teste, através de geradores de ruído uniforme, para proceder ao teste dos parâmetros estáticos do conversor. Em termos de parâmetros estáticos, verificaram-se melhores resultados com a utilização dos valores gerados através do gerador uniforme Mersenne Twister, utilizados para testar o MAX189, que com os valores gerados através do gerador uniforme Congruente Multiplicativo, utilizados para testar o ADC do Arduino Due. No entanto, os valores da INL para o teste com o gerador Mersenne Twister encontram-se fora do intervalo esperado. Tal pode dever-se ao número de amostras utilizadas para teste, poderia ser necessário um maior número de amostras, e à utilização do sinal de PWM para enviar os valores para o ADC, a oscilação deste sinal pode ter influenciado os valores dos parâmetros estáticos. Relativamente ao ADC do Arduino Due, o teste efetuado apresenta resultados muito diferentes dos resultados típicos apresentados pelo fabricante. Isto significa que o ADC tem um desempenho superior ao esperado.

O trabalho desenvolvido nesta dissertação foi utilizado para a elaboração de um *poster* e dois artigos. O *poster* foi desenvolvido no âmbito da conferência CETC 2016[36], sob o título *"The Interface and Control System of the Upgraded HVOpto/HVRemote Card of the TileCal"*, tendo este *poster* recebido o prémio de melhor *poster*, o resumo do *poster* pode ser consultado em [53]. Ainda no âmbito desta conferência, foi desenvolvido um artigo para a revista i-ETC[37], que já foi aprovado e aguarda publicação. Em Junho, foi submetido um artigo para a conferência DCIS2017[38]. Este artigo, intitulado de *"A Simple and Inexpensive Platform for ADC Histogram Testing"*, apresenta a interface em *Python* utilizada para testar os ADCs e os respetivos resultados.

[36] Sigla de *Conference On Electronics, Telecommunications and Computers*, patrocinada pelo ISEL.
[37] Revista *ISEL Academic Journal Of Electronics Telecommunications And Computers*.
[38] Sigla de *Design Of Circuits and Integrated Systems*.

Ainda existe muito trabalho a desenvolver futuramente. O teste dos DACs não foi efetuado devido aos erros no 1º protótipo da placa de controlo e ao estado atual do 2º protótipo. Apesar de se terem conseguido obter amostras do MAX189, com funcionamento muito semelhante ao TLV2541, não foi possível obter amostras de DACs semelhantes ao DAC7568 nem ao MAX5725. Esta foi a principal razão pela qual não foi testado nenhum DAC, pois uma montagem experimental semelhante à utilizada para o MAX189 poderia ter sido utilizada. O teste dos DACs será efetuado assim que forem reunidas condições para tal. Também não foi realizado nenhum teste com um gerador de ruído Gaussiano, como tinha sido previsto inicialmente. Este teste ainda será realizado, inclusive com os DACs. O 2º protótipo da placa de controlo ainda não está em condições de teste. Assim que a assemblagem dos componentes for efetuada, serão efetuados os devidos testes para verificar o funcionamento da placa bem como dos seus componentes.

Para a HV Remote, deverá ser realizada uma expansão da interface de utilizador que permita o seu funcionamento com todos os componentes da placa e a comunicação através do módulo da Tibbo, EM1206+RJ203, mencionado no Capítulo 2.

7 BIBLIOGRAFIA

[1] G. F. Giudice, "Prologue," em *A Zeptospace Odyssey - A Journey into the Physics of the LHC*, Oxford University Press, 2009.

[2] C. O'Luanaigh, "A vacuum as empty as interstellar space," CERN, 31 Agosto 2012. [Online]. Available: https://home.cern/about/engineering/vacuum-empty-interstellar-space. [Acedido em 2 Julho 2016].

[3] C. O'Luanaigh, "The Large Hadron Collider," CERN, 21 Janeiro 2014. [Online]. Available: https://home.cern/topics/large-hadron-collider. [Acedido em 2 Julho 2016].

[4] F. Tang, K. Anderson, G. Drake, J.-F. Genat, M. Oreglia, J. Pilcher e L. Price, "Design of the Front-End Readout Electronics for ATLAS Tile Calorimeter at the sLHC," *IEEE Transactions On Nuclear Science*, vol. 60, nº 2, pp. 1255-1259, 2013.

[5] R. Leitner, "Status of the ATLAS Hadronic Tile Calorimeter," em *3rd International Workshop on Very High multiplicity Physics*, Dubna, 2002.

[6] L. Plazak, "The ATLAS calorimeter at the LHC and the phase II upgrade program," em *XXIInd International Workshop "High-Energy and Quantum Field Theory"*, Samara, Russia, 2015.

[7] P. Vankov, "ATLAS Upgrade for the HL-LHC: meeting the challenges of a five-fold increase in collision rate," em *EPJ Web of Conferences - Hadron Collider Physics symposium*, 2011.

[8] S. Muschter, H. Aakerstedt, K. Anderson, C. Bohm, M. Oreglia e F. Tang, "Development of a digital readout board for the ATLAS Tile Calorimeter upgrade demonstrator," *Journal of Instrumentation*, vol. 9, 2014.

[9] J. D. Alves, "Interface Ethernet para um Testador de Sistemas Electrónicos do Tilecal," Departamento de Física da Faculdade de Ciências da Universidade de Lisboa, Lisboa, 2012.

[10] J. A. T. Pina, "The Control System of the ATLAS/TILECAL," Departamento de Física da Faculdade de Ciências da Universidade de Lisboa, Lisboa, 2010.

[11] F. Vazeille, "Performance of a Remote High Voltage Power Supply for the Phase II Upgrade of the ATLAS Tile Calorimeter," CERN, Geneva, 2015.

[12] "8-Channel Level Translators," Maxim Integrated, 2008.

[13] "16-Bit I/O Expander with Serial Interface," Microchip Technology Inc., 2007.

[14] "12-/14-16-Bit, Octal Channel, Ultralow Glitch, Voltage Output Digital-to-Analog Converters with 2.5, 2ppm/ºC Internal Reference," Texas Instruments, 2009, Revised 2014.

[15] "Serial Analog-To-Digital Converters with Autopower Down," Texas Instruments Incorporated, 2000, Revised 2010.

[16] "REF02, +5V Precision Voltage Reference," Burr-Brown Produts from Texas Instruments, 1993, Revised 2005.

[17] "Low Cost, Current Output Temperature Transducer," Analog Devices, Norwood, USA, 2003.

[18] "2 - Terminal IC 1.2 V Reference," Analog Devices, Norwood, USA, 2004.

[19] "Single-Ended 16-Channel/Differential 8-Channel CMOS Analog Multiplexers," Burr-Brown Products from Texas Instruments, 1988, Revised 2003.

[20] "INA 12x Precision, Low Power Instrumentation Amplifiers," Texas Instruments, 1995, Revised 2015.

[21] I. Maxim Integrated Products, "Ultra-Small, Octal-Channel, 8-/10-/12-Bit Buffered Output DACs with Internal Reference and SPI Interface," Maxim Integrated, 2013.

[22] H. Jiang, B. Olleta, D. Chen e R. L. Geiger, "Testing High-Resolution ADCs With Low-Resolution/Accuracy Deterministic Dynamic Element Matched DACs," *IEEE Transactions on Instrumentation and Measurement*, vol. 56, nº 5, pp. 1753-1762, Outubro 2007.

[23] A. Lechner e A. Richardson, "Test of A/D Converters, From converter characteristics to built-in self-test proposals," em *Test and Diagnosis of Analogue, Mixed-Signal and RF Integrated Circuits: The System on Chip Approach*, University of Lancaster, UK, 2008, p. 289.

[24] E. Korhonen, "On-Chip Testing of A/D and D/A Converters, Static Linearity Testing Without Statistically Known Stimulus," Faculty of Technology, Department of Electrical and Information Engineering, University of Oulu;, Oulu, 2010.

[25] K. H. Lundberg, "Analog-to-Digital Converter Testing," MIT, 2002.

[26] G. Evans, "Geradores de Ruído Branco Gaussiano e Uniforme para a Realização de Teste Automático de ADC's em Circuitos Integrados CMOS," 2006.

[27] R. C. Martins e A. C. Serra, "ADC characterization by using the histogram test stimulated by Gaussian noise, Theory and experimental results," *Elsevier,* pp. 291-300, 2000.

[28] S. J. Upadhyaya, "Noise Generators," Department of Electrical & Computer Engineering, State University of New York at Buffalo, Buffalo, NY 14260, 1998.

[29] N. Kularatna, Electronic Circuit Design: From Concept to Implementation, CRC Press, 2008.

[30] A. Demir e A. Sangiovanni-Vincentelli, Analysis and Simulation of Noise in Nonlinear Electronic Circuits and Systems, Springer Science & Business Media, 2012.

[31] D. B. Thomas, W. Luk, P. H. W. Leong e J. D. Villasenor, "Gaussian Random Number Generators," *ACM Computing Surveys,* vol. 39, nº 4, October 207.

[32] G. Box e M. Muller, "A note on the generation of random normal deviates," *Ann. Math. Statist,* vol. 29, nº 2, pp. 610-611, 1958.

[33] I. Florescu, Probability and Stochastic Processes, John Wiley & Sons, 2014.

[34] S. Bandyopadhyay e R. Bhattacharya, Discrete and Continuous Simulation: Theory and Practive, CRC Press, 2014.

[35] J. E. Gentle, Statistics and Computing: Random Number Generation and Monte Carlo Methods, Springer Science & Business Media, 2013.

[36] G. Marsaglia e W. W. Tsang, "The Ziggurat Methot for Generating Random Variables," 2000.

[37] S. L. Anderson, "Random Number Generators on Vector Supercomputers and Other Advanced Architectures," *SIAM Review,* vol. 2, pp. 221-251, Jun 1990.

[38] G. Marsaglia e A. Zaman, "A New Class of Random Number Generators," *The Annals of Applied Probability,* vol. 1, nº 3, pp. 462-480, 1991.

[39] A. B. Orue, F. Montoya e L. H. Encinas, "Trifork, a New Pseudorandom Number Generator Based on Lagged Fibonacci Maps," *Journal of Computer Science and Engineering,* 2010.

[40] A. Jagannatam, "Mersenne Twister - A Pseudo Random Number Generator and its Variants," The Pennsylvania State University CiteSeerX Archives.

[41] M. A. Zidan, A. G. Radwan e K. N. Salama, "Random Number Generation Based on Digital Differential Chaos," em *IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Seoul, 2011.

[42] S. Robson, B. Leung e G. Gong, "Truly Random Number Generator Based on a Ring Oscillator Utilizing Last Passage Time," *IEEE Transactions on Circuits and Systems*, vol. 61, nº 12, December 2014.

[43] F. Leens, “An Introduction to I2C and SPI Protocols,” *IEEE Instrumentation & Measurement Magazine*, Fevereiro 2009.

[44] M. Usach, “SPI Interface, Application Note AN- 1248,” Analog Devices, Norwood.

[45] “Arduino - Communication,” Tutorials Point, 2016. [Online]. Available: https://www.tutorialspoint.com/arduino/arduino_communication.htm. [Acedido em 5 Dezembro 2016].

[46] C. Buffington, “MCP23S17 Class for Arduino,” Março 2012. [Online]. Available: http://playground.arduino.cc/Main/MCP23S17. [Acedido em Novembro 2016].

[47] Arduino, “analogWrite(),” Arduino, [Online]. Available: https://www.arduino.cc/en/Reference/analogWrite. [Acedido em 25 Abril 2017].

[48] Arduino, “PWM,” Arduino, [Online]. Available: https://www.arduino.cc/en/Tutorial/PWM. [Acedido em 25 Abril 2017].

[49] Arduino, “Adjusting PWM Frequencies,” Arduino, [Online]. Available: http://playground.arduino.cc/Main/TimerPWMCheatsheet. [Acedido em 25 Abril 2017].

[50] J. D. Alves e G. Evans, “Digital Pseudorandom Uniform Noise Generators for ADC Histogram Test,” em *DCIS 2015 - XXX Conference on Design of Circuits and Integrated Systems*, Estoril, Lisboa, 2015.

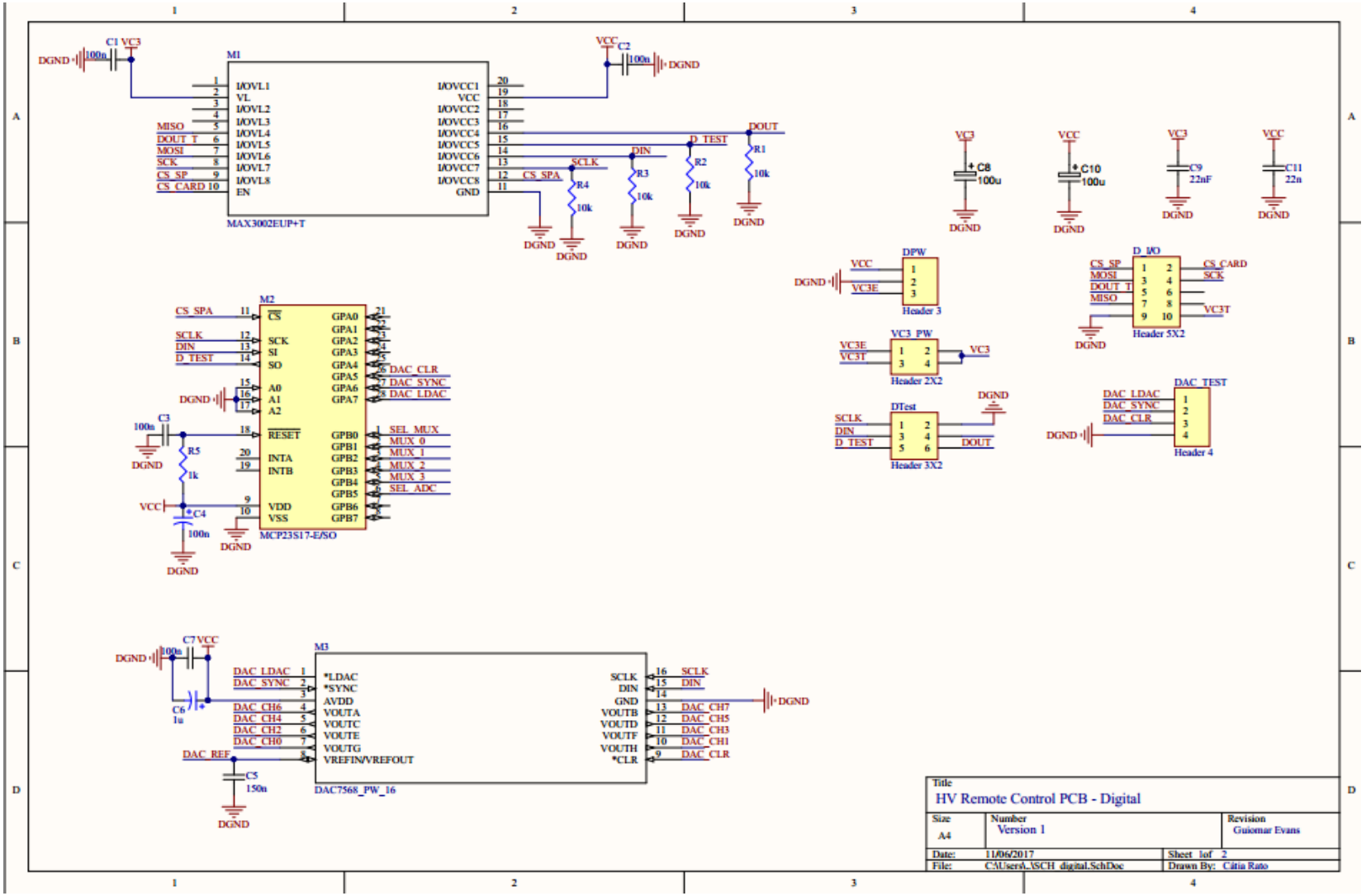
[51] M. Integrated, “MAX187/MAX189 +5 V, Low-Power, 12-Bit Serial ADCs,” Maxim Integrated Products, Inc., 2012.

[52] Atmel, “SAM3X/SAM3A Series - Atmel|Smart ARM-Based MCU,” Atmel Corporation, 2015.

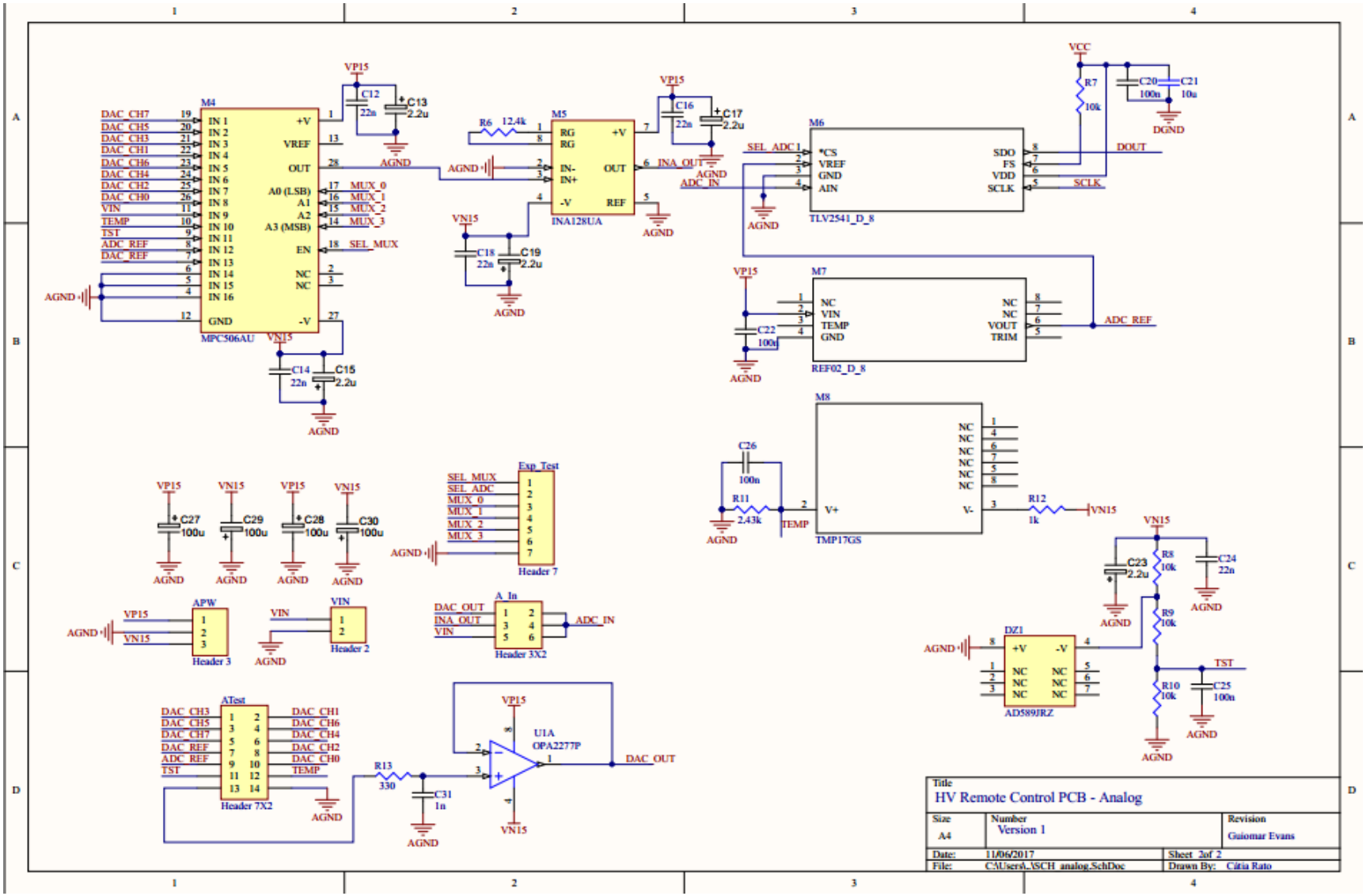
[53] C. Rato, J. Sabino, L. Gurriana, L. Seabra, A. Gomes, G. Evans, J. S. Augusto e A. Maio, “The Interface and Control System of the Upgraded HVOpto/HVRemote Card of the TileCal,” FCUL, LIP, BioISI, Inesc-ID, 7 Dezembro 2016. [Online]. Available: http://cetc2016.isel.pt/resources/program/Poster%201/Poster1_4.pdf.

8 ANEXOS

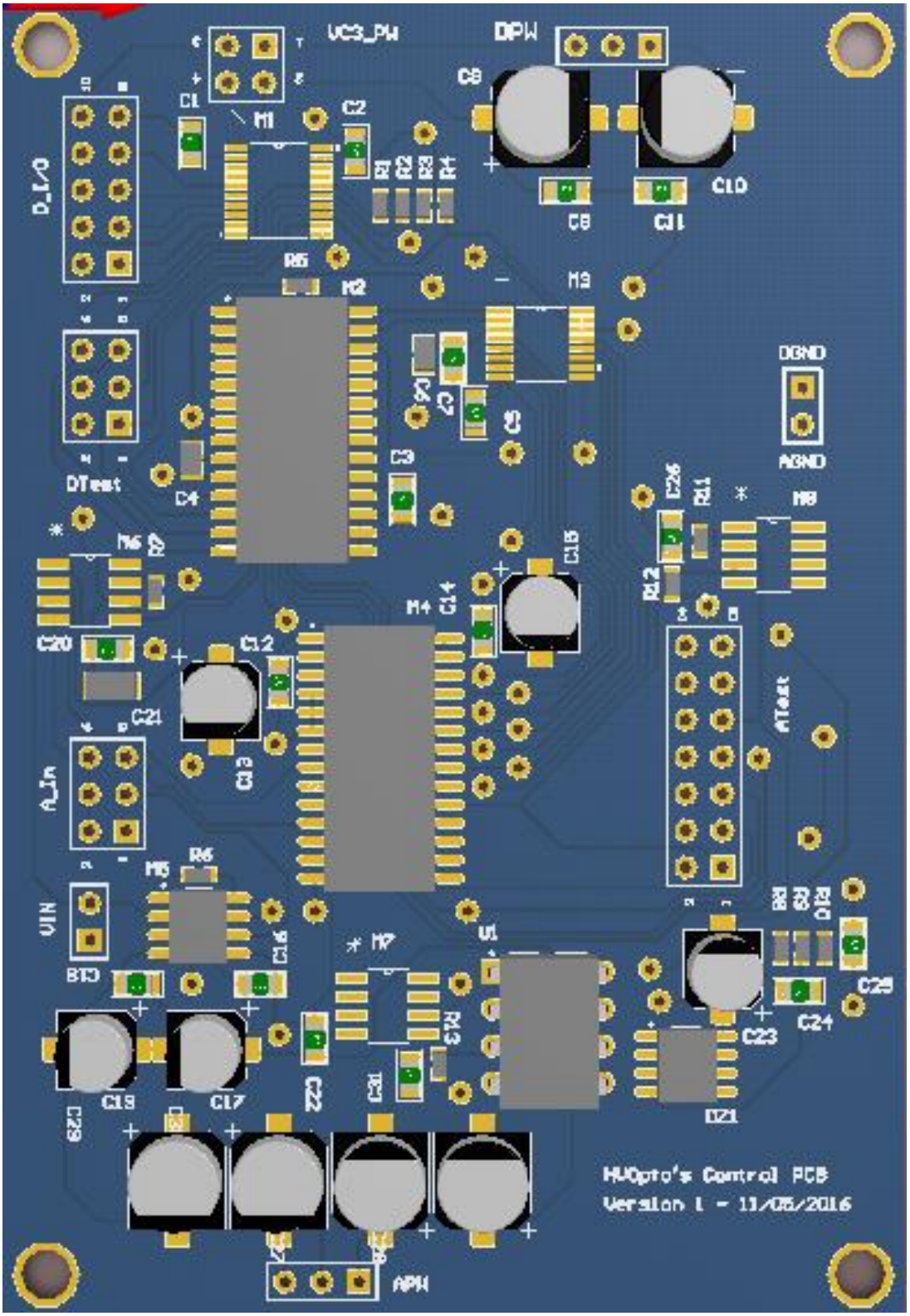
8.1 ESQUEMA ELÉTRICO DA PARTE DIGITAL DA 1ª VERSÃO DA PLACA DE CONTROLO



8.2 ESQUEMA ELÉTRICO DA PARTE ANALÓGICA DA 1ª VERSÃO DA PLACA DE CONTROLO



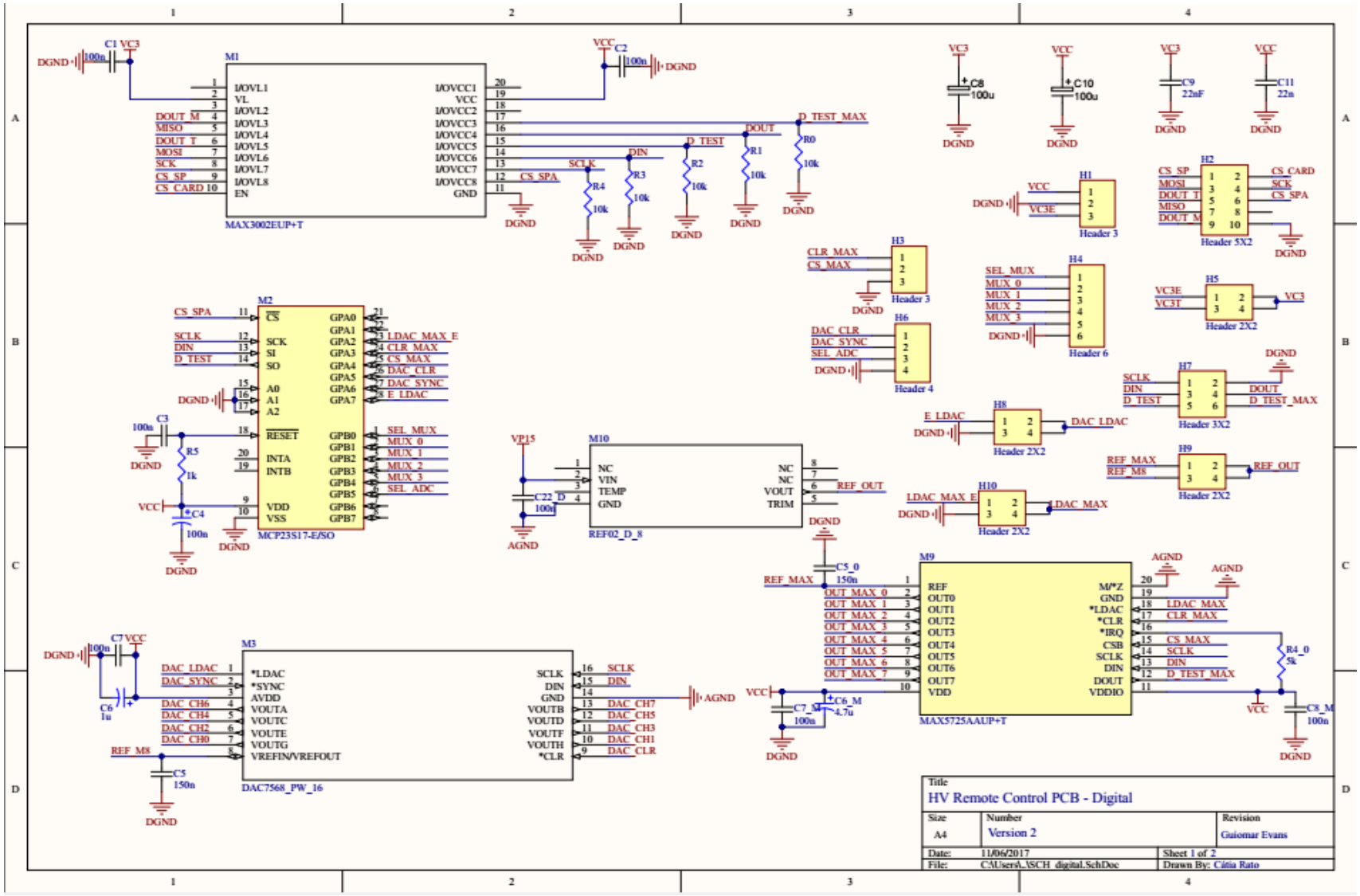
8.3 Diagrama do PCB da 1ª Versão da Placa de Controlo da HV Remote



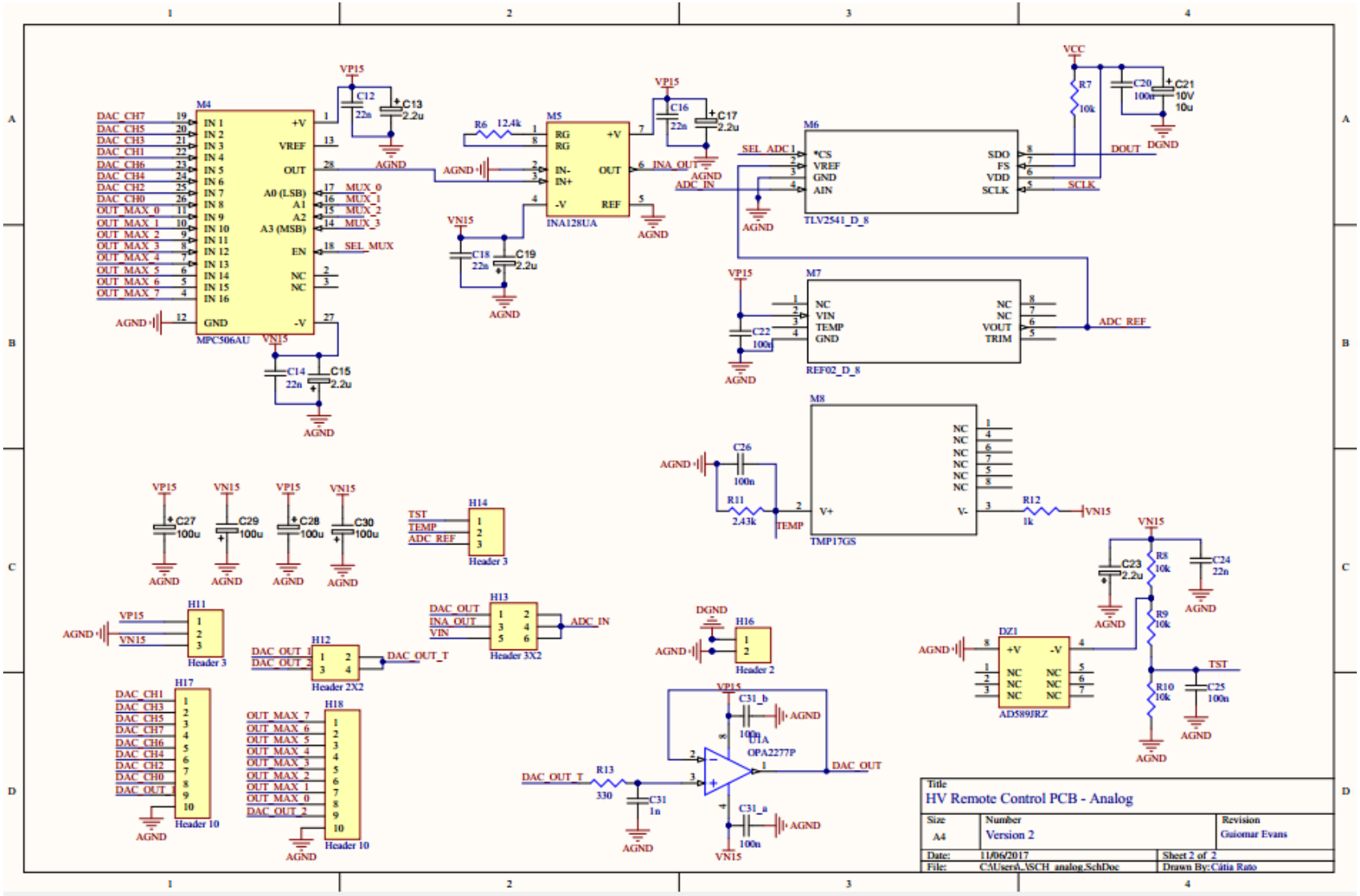
8.4 LISTA DE MATERIAL DA 1ª VERSÃO DA PLACA DE CONTROLO

Componente	Referência	Quantidade	Encapsulamento
Conector 5x2	Ficha para a placa Tibbo	1	DIP
Conector 5x2	Ficha para o *header* D_I/O	2	DIP
Conector 3x2	Ficha para o *header* DTest, A_In	2	DIP
Conector 7x2	Ficha para o *header* ATest	1	DIP
Conector 1x2	Ficha para o *header* VIN	1	DIP
Conector 2x2	Ficha para o *header* VC3_PW	1	DIP
Conector 3 pinos	Fichas para os *headers* DPW e APW	2	DIP
Header 5x2	D_I/O	1	DIP
Header 3x2	DTest, A_In	2	DIP
Header 7x2	ATest	1	DIP
Header 1x2	VIN	1	DIP
Header 2x2	VC3_PW	1	DIP
Header 3 pinos	*Headers* DPW e APW	2	DIP
MAX3002	M1	2	TSSOP (20 pinos)
MCP23S17	M2	2	SOIC (28 pinos)
DAC7568C	M3	2	TSSOP (16 pinos)
MCP506A	M4	2	SOIC (28 pinos)
INA128	M5	2	SOIC (8 pinos)
TLV2541	M6	2	SOIC (8 pinos)
REF02	M7	2	SOIC (8 pinos)
TMP17	M8	2	DIP (8 pinos)
AD589	DZ1	2	SOIC (8 pinos)
OPA2277P	U1A	2	DIP (8 pinos)
R-1k, 250 mW, 1%	R5, R12	2	0603
R-2.43k, 250 mW, 1%	R11	1	0603
R-10k, 250 mW, 1%	R1, R2, R3, R4, R7, R8, R9, R10	8	0603
R-12,4k, 250mW, 1%	R6	1	0603
C-1n, 25V, 10%	C31	1	0805
C-22n, 25 V, 10%	C9, C11, C12, C14, C16, C18, C24	7	0805
C-100n, 25 V, 10%	C1, C2, C3, C4, C7, C20, C22, C25, C26	9	0805
C-150n, 25 V, 10%	C5	1	0805
C-1u, 25 V, 10%	C6	1	0805
C-2.2u, 25 V, 10%	C13, C15, C17, C19, C23	5	1206
C-10u, 25 V, 10%	C21	1	1206
C-100u, 25 V, 10%	C8, C10, C27, C28, C29, C30	6	1210

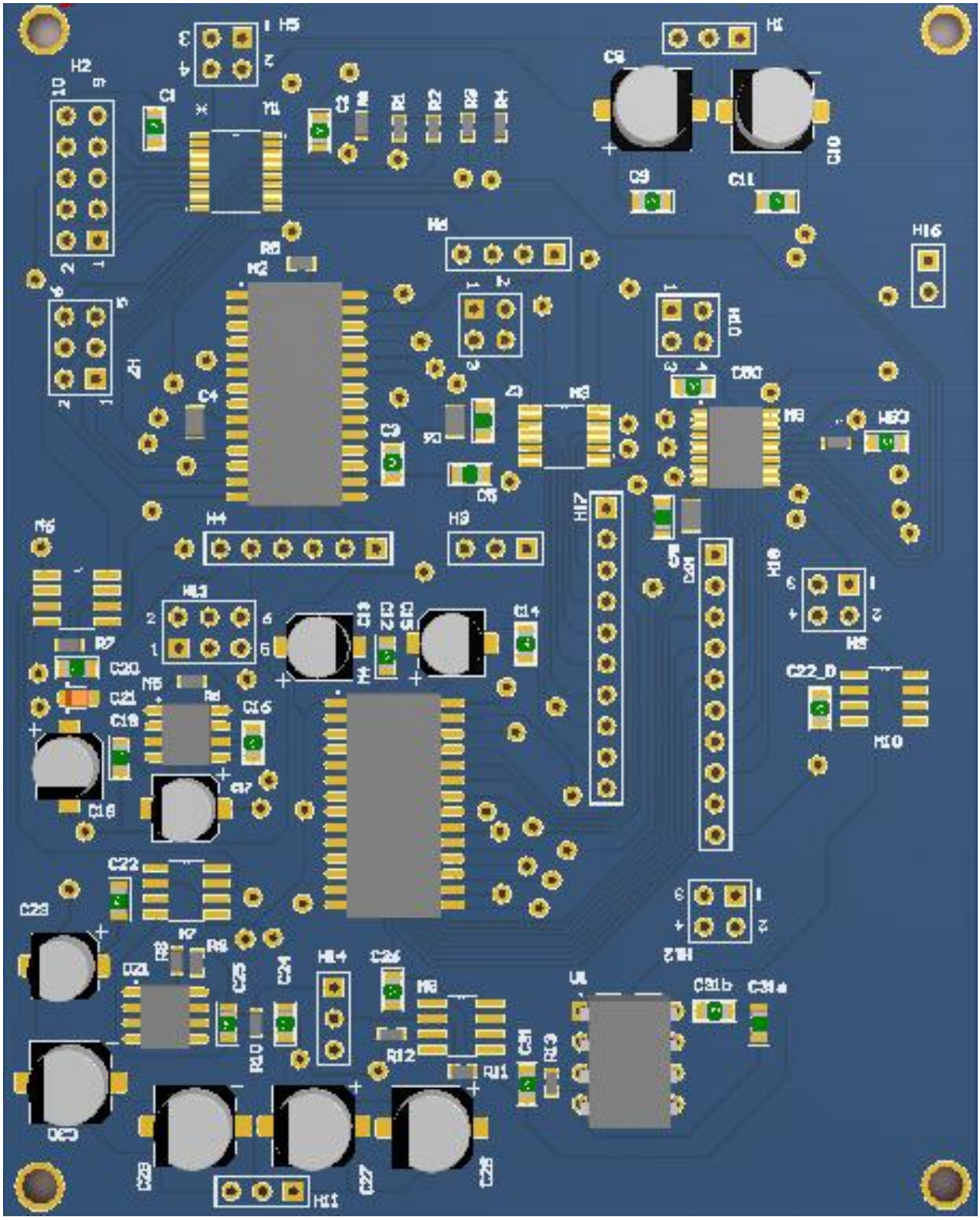
8.5 ESQUEMA ELÉTRICO DA PARTE DIGITAL DA 2ª VERSÃO DA PLACA DE CONTROLO



8.6 ESQUEMA ELÉTRICO DA PARTE ANALÓGICA DA 2ª VERSÃO DA PLACA DE CONTROLO



8.7 Diagrama do PCB da 2ª Versão da Placa de Controlo da HV Remote



8.8 LISTA DE MATERIAL DA 2ª VERSÃO DA PLACA DE CONTROLO

Componente	Referência	Quantidade	Encapsulamento
Conector 5x2	Ficha para a placa Tibbo	1	DIP
Conector 5x2	Ficha para o *header* D_I/O	2	DIP
Conector 3x2	Ficha para o *header* DTest, A_In	2	DIP
Conector 7x2	Ficha para o *header* ATest	1	DIP
Conector 1x2	Ficha para o *header* VIN	1	DIP
Conector 2x2	Ficha para o *header* VC3_PW	1	DIP
Conector 3 pinos	Fichas para os *headers* DPW e APW	2	DIP
Header 5x2	D_I/O	1	DIP
Header 3x2	DTest, A_In	2	DIP
Header 7x2	ATest	1	DIP
Header 1x2	VIN	1	DIP
Header 2x2	VC3_PW	1	DIP
Header 3 pinos	*Headers* DPW e APW	2	DIP
MAX3002	M1	2	TSSOP (20 pinos)
MCP23S17	M2	2	SOIC (28 pinos)
DAC7568C	M3	2	TSSOP (16 pinos)
MCP506A	M4	2	SOIC (28 pinos)
INA128	M5	2	SOIC (8 pinos)
TLV2541	M6	2	SOIC (8 pinos)
REF02	M7	2	SOIC (8 pinos)
TMP17	M8	2	DIP (8 pinos)
AD589	DZ1	2	SOIC (8 pinos)
OPA2277P	U1A	2	DIP (8 pinos)
R-1k, 250 mW, 1%	R5, R12	2	0603
R-2.43k, 250 mW, 1%	R11	1	0603
R-10k, 250 mW, 1%	R1, R2, R3, R4, R7, R8, R9, R10	8	0603
R-12,4k, 250mW, 1%	R6	1	0603
C-1n, 25V, 10%	C31	1	0805
C-22n, 25 V, 10%	C9, C11, C12, C14, C16, C18, C24	7	0805
C-100n, 25 V, 10%	C1, C2, C3, C4, C7, C20, C22, C25, C26	9	0805
C-150n, 25 V, 10%	C5	1	0805
C-1u, 25 V, 10%	C6	1	0805
C-2.2u, 25 V, 10%	C13, C15, C17, C19, C23	5	1206
C-10u, 25 V, 10%	C21	1	1206
C-100u, 25 V, 10%	C8, C10, C27, C28, C29, C30	6	1210

8.9 CÓDIGO DA INTERFACE UTILIZADOR EM *PYTHON* PARA O TESTE DO EXPANSOR MCP23S17

```
from tkinter import *
from tkinter import ttk
from time import sleep
import serial
#ser = serial.Serial('COM4', 9600, timeout = 0)
#Parte gráfica da janela de teste
win = Tk()
win.title("Serial to Paralel Expander Test")
frame = ttk.Frame(win, padding = "3 3 12 12")
frame.grid(column = 0, row = 0, sticky = (N, W, E, S))
frame.columnconfigure(0, weight = 1)
frame.rowconfigure(0, weight = 1)
win.geometry('550x200')
#Configuração do expansor, modo leitura (escrevemos para os portos A e B)
Message(win, text = "Testing MCP23S17", width = 200).grid(column = 1, row = 1)

Label(win, text = "Port A").grid(column = 2, row = 3)
Label(win, text = "Port B").grid(column = 4, row = 3)

#Dados enviados para o circuito
data_sent1 = Entry(win, width = 10)
data_sent1.grid(column = 2, row = 4)
data_sent2 = Entry(win, width = 10)
data_sent2.grid(column = 4, row = 4)

file_data = open('Expander_Data.txt', 'a')
file_data.write('Sent\t\t\t\t\tReceived\n')

#Os dados a enviar são binários
def send_data():
    data1 = []
    data2 = []
    data1 = data_sent1.get()
    data2 = data_sent2.get()
    data1 = str(data1)
    data2 = str(data2)
    #ser.write(data1.encode())
    sleep(100)
    #ser.write(data2.encode())
    sleep(10)
    file_data.write(data1 + ' ' + data2 + '\t\t')

#send_data corresponde ao DIN
Label(win, text = "").grid(column = 3, row = 4)
Label(win, text = "").grid(column = 3, row = 5)

#Dados recebidos do circuito , vão corresponder ao D_TEST
```

```
data_received1 = Entry(win, width = 10)
data_received1.grid(column = 2, row = 5)
data_received2 = Entry(win, width = 10)
data_received2.grid(column = 4, row = 5)

def recv_data():
    arr = []
    arr1 = ""
    while(len(arr) < 5):
        #data = ser.readline().decode()
        #arr.append(data)
        arr = [i.strip('\r\n') for i in arr]
        sleep(1)
    arr1 = arr[0]
    data1 = []
    data2 = []
    for i in range(0, 8):
        data1.append(arr1[i])
    for i in range(8, 16):
        data2.append(arr1[i])
    data1 = ''.join(x for x in data1)
    data2 = ''.join(x for x in data2)
    data_received1.insert(INSERT, data1)
    data_received2.insert(INSERT, data2)
    received_data = data1 + data2
    file_data.write(data1 + ' ' + data2 + '\n')
    file_data.close()
    #ser.close()
    return received_data

Label(win, text = "Paralel Output Data").grid(column = 1, row = 4)
Label(win, text = "Register Data").grid(column = 1, row = 5)

Button(win, text = "Send", command = send_data).grid(column = 5, row = 4)
Button(win, text = "Get", command = recv_data).grid(column = 5, row = 5)
Button(win, text = "Exit", command = win.destroy).grid(column = 3, row = 6)

for child in win.winfo_children():
    child.grid_configure(padx = 5, pady = 5)
```

8.10 CÓDIGO DA INTERFACE UTILIZADOR EM *PYTHON* PARA O TESTE DO DAC E DO ADC

```
from tkinter import *
from tkinter import ttk
from tkinter import filedialog
from time import sleep
import serial
import os.path
import csv

#Parte gráfica da janela de teste
win = Tk()
win.title("A/D and D/A Converters Test")
frame = ttk.Frame(win, padding = "3 3 12 12")
frame.grid(column = 0, row = 0, sticky = (N, W, E, S))
frame.columnconfigure(0, weight = 1)
frame.rowconfigure(0, weight = 1)
win.geometry('360x240')


Label(win, text = "").grid(column = 0)
Label(win, text = "").grid(column = 2)
Label(win, text = "").grid(column = 4)
Label(win, text = "").grid(row = 12)
Label(win, text = "").grid(row = 0)

Label(win, text = "Select Test Type").grid(column = 1, row = 1)
testtype = IntVar()
testtype_list = []
testtype_name = ['DAC', 'ADC', 'DAC + ADC']

#Seleccionar tipo de teste
for i in range(3):
   testtype_list.insert(i,testtype_name[i])
testlist = ttk.Combobox(win, width = 10, state = "readonly",
                        values = testtype_list,
                        textvariable = testtype)
testlist.grid(column = 3, row = 1)
testtype = ' '
def get_testtype():
   global testtype
   testtype = testlist.get()
   return(testtype)

#Criar botões: escolher ficheiro
selected_file = Entry(win)
selected_file.grid(column = 3, row = 3)

#Ficheiro onde são guardados os dados
```

```
Label(win, text = "Values saved on:").grid(column = 1, row = 7)
value_sent = Entry(win)
value_sent.grid(column = 3, row = 7)

#Contagem de dados
Label(win, text = "Data Count").grid(column = 1, row = 9)
data_count = Entry(win)
data_count.grid(column = 3, row = 9)

#Contagem temporal
Label(win, text = "Time Elapsed").grid(column = 1, row = 11)
time_elapsed = Entry(win)
time_elapsed.grid(column = 3, row = 11)

#Aparece janela pop up, que permite selecionar o ficheiro a ler
filename = ''
def openDataFile():
    global filename
    filename = filedialog.askopenfilename(initialdir = "/", title = "Select File")
    selected_file.insert(INSERT,os.path.basename(filename))
#save_data = open("DAC_Data.txt",'a')

#Selecionar canais do DAC
Label(win, text = "DAC Channel").grid(column = 1, row = 5)
channel = IntVar()
channel_list = []
for i in range(8):
    channel_list.insert(i,i)
ch_list = ttk.Combobox(win, width = 10, state = "readonly",
                       values = channel_list,
                       textvariable = channel)
ch_list.grid(column = 3, row = 5)
channel = ' '
def get_channel():
    global channel
    channel = ch_list.get()
    return(channel)

def start_test():
    if(testtype==1):
        send_data_dac()
    elif(testtype==2):
        send_data_adc()
    elif(testtype==3):
        send_data_adc_dac()

def send_data_dac():
    count = 0
    text_file = "DAC_Data.txt"
    open(text_file,'w').close()
    #ser = serial.Serial('COM3', 9600, timeout = 0)
```

```
sleep(10)
test_voltage = []
readFile = open(filename, 'r')
for line in readFile.readlines():
    for i in line.split(","):
        test_voltage.append(int(i))
start = time.time()
for i in test_voltage:
    arr = []
    data1 = []
    bin_voltage = format(i,'b').zfill(16)
    value_sent.insert(INSERT, bin_voltage)
    #DAC channel 0
    #Canal 9 do mux
    if(channel == 0):
        data = '0000000010111000' + '000000000110' + bin_voltage + '0000'
        #ser.write(data.encode())
        sleep(.1)
    #DAC channel 1
    #Canal 4 do mux
    elif(channel == 1):
        data = '0000000010011000' + '000000000111' + bin_voltage + '0000'
        #ser.write(data.encode())
        sleep(.1)
    #DAC channel 2
    #Canal 7 do mux
    elif(channel == 2):
        data = '0000000010110000' + '000000000100' + bin_voltage + '0000'
        #ser.write(data.encode())
        sleep(.1)
    #DAC channel 3
    #Canal 3 do mux
    elif(channel == 3):
        data = '0000000010010000' + '000000000101' + bin_voltage + '0000'
        #ser.write(data.encode())
        sleep(.1)
    #DAC channel 4
    #Canal 6 do mux
    elif(channel == 4):
      data = '0000000010101000' + '000000000010' + bin_voltage + '0000'
      #ser.write(data.encode())
      sleep(.1)
    #DAC channel 5
    #Canal 2 do mux
    elif(channel == 5):
        data = '0000000010001000' + '000000000011' + bin_voltage + '0000'
        #ser.write(data.encode())
        sleep(.1)
    #DAC channel 6
    #Canal 5 do mux
    elif(channel == 6):
```

```
            data = '0000000010100000' + '000000000000' + bin_voltage + '0000'
            #ser.write(data.encode())
            sleep(.1)
        #DAC channel 7
        #Canal 1 do mux
        elif(channel == 7):
            data = '0000000010000000' + '000000000001' + bin_voltage + '0000'
            #ser.write(data.encode())
            sleep(.1)
        while(len(arr) < 33):
            #data = ser.readline().decode()
            #arr.append(data)
            arr = [i.strip('\r\n') for i in arr]
            sleep(.1)
        #verificar o início do ciclo for
        arr = ''.join(x for x in arr)
        for i in range(0, len(arr)-1):
            data1.append(arr[i])
        data1 = ''.join(x for x in data1)
        save_data = open(text_file,'a')
        save_data.write(data1 + ",")
        count = count + 1
    data_count.inser(INSERT,count)
    end = time.time()
    time_elapsed.insert(INSERT,round(end-start,2))
    save_data.close()
    #ser.close()

def send_data():
    text_file = "adc_data_MT.txt"
    open(text_file,'w').close()
    count = 0
    arr = []
    ser = serial.Serial('COM3',100000,timeout = 0)
    sleep(1)
    test_voltage = []
    readFile = open(filename, 'r')
    for line in readFile.readlines():
        for i in line.split(","):
            test_voltage.append(i)
    #print("\nStarting to send values!\n")
    #Primeiros 8 bits correspondem à configuração do mux
    #São definidos no script do arduino
    start = time.time()
    for i in test_voltage:
        #print("send!")
        data_sent = float(i)
        data_sent = str(data_sent*51)
        print(float(i))
        ser.write(data_sent.encode())
        ##print("Collecting data...")
```

```
        sleep(1)
        while(len(arr) < 1):
            a = ser.readline().decode()
            arr.append(a)
        data_received = arr[0]
        print(data_received)
        save_data = open(text_file,'a')
        save_data.write(data_received + ",")
        arr = []
        count = count + 1
        print(count)
    data_count.insert(INSERT,count)
    value_sent.insert(INSERT,text_file)
    end = time.time()
    time_elapsed.insert(INSERT,round(end-start,2))
    save_data.close

def send_data_adc_dac():
    return 1

#Botões
Button(win, text = "Select Data File", command = openDataFile).grid(column = 1, row = 3) #Escolher
o ficheiro
Button(win, text = "Ok", command = get_testtype).grid(column = 5, row = 1) #Guardar o tipo de teste
Button(win, text = "Start Data Transfer", command = start_test).grid(column = 5, row = 3) #Começar o
Teste
Button(win, text = "Ok", command = get_channel).grid(column = 5, row = 5) #Guardar o canal
Button(win, text = "Exit", command = win.destroy).grid(column = 3, row = 13) #Fechar Janela
```

8.11 Código da Interface Utilizador em *Python* para os Testes de Desempenho

```
from tkinter import *
from tkinter import ttk
import visa
import os
import struct
import socket
from time import sleep
import serial

win3 = Tk()
win3.title("Performance Tests")
cframe = ttk.Frame(win3, padding = "3 3 12 12")
cframe.grid(column = 0, row = 0, sticky = (N, W, E, S))
cframe.columnconfigure(0, weight = 1)
cframe.rowconfigure(0, weight = 1)
win3.geometry('550x350')

#Definição de variáveis
tst = DoubleVar()
temp1v = DoubleVar()
temp2v = DoubleVar()
temp1c = DoubleVar()
temp2c = DoubleVar()
adc_ref = DoubleVar()
dac_ref= DoubleVar()
dac2_ref= DoubleVar()
ck = DoubleVar()

#Definição de entries
tst_entry = Entry(win3, width = 16)
tst_entry.grid(column = 2, row = 3)
temp1v_entry = Entry(win3, width = 16)
temp1v_entry.grid(column = 2, row = 5)
temp1c_entry = Entry(win3, width = 16)
temp1c_entry.grid(column = 4, row = 5)
adc_ref_entry = Entry(win3, width = 16)
adc_ref_entry.grid(column = 2, row = 7)
dac_ref_entry = Entry(win3, width = 16)
dac_ref_entry.grid(column = 2, row = 9)
dac2_ref_entry = Entry(win3, width = 16)
dac2_ref_entry.grid(column = 2, row = 11)
ck_entry = Entry(win3, width = 16)
ck_entry.grid(column = 2, row = 13)

os_e = Entry(win3, width = 30)
os_e.grid(column = 2, row = 1)
```

```
#Definir esta função para osciloscópio e multímetro
#Tentar detectar o tipo de dispositivo ligado e seleccioná-lo
def get_address():
    rm = visa.ResourceManager()
    rm_list = rm.list_resources()
    os_address = rm_list[0]
    os_e.insert(INSERT, os_address)
    scope = rm.open_resource(os_address)
    return scope

Button(win3, text = "Get Reading Device Address", command = get_address).grid(column = 1, row =
1)

#Funções abaixo têm que ser todas alteradas, de acordo com o valor devolvido por get_address
def get_tst():
    rm = visa.ResourceManager()
    scope = rm.open_resource('USB0::0x0699::0x036A::C011565::INSTR')
    tst = scope.query("MEASUrement:MEAS1:VALue?")
    tst_entry.insert(INSERT, round(float(tst),3))

def get_temp1():
    rm = visa.ResourceManager()
    scope = rm.open_resource('USB0::0x0699::0x036A::C011565::INSTR')
    temp1v = scope.query("MEASUrement:MEAS1:VALue?")
    temp1v_entry.insert(INSERT, round(float(temp1v),3))
    #temp1c

def get_temp2():
    rm = visa.ResourceManager()
    scope = rm.open_resource('USB0::0x0699::0x036A::C011565::INSTR')
    temp2v = scope.query("MEASUrement:MEAS1:VALue?")
    temp2v_entry.insert(INSERT, round(float(temp2v),3))
    #temp2c

def get_adc_ref():
    rm = visa.ResourceManager()
    scope = rm.open_resource('USB0::0x0699::0x036A::C011565::INSTR')
    adc_ref = scope.query("MEASUrement:MEAS1:VALue?")
    adc_ref_entry.insert(INSERT, round(float(adc_ref),3))

def get_dac_ref():
    rm = visa.ResourceManager()
    scope = rm.open_resource('USB0::0x0699::0x036A::C011565::INSTR')
    dac_ref = scope.query("MEASUrement:MEAS1:VALue?")
    dac_ref_entry.insert(INSERT, round(float(dac_ref),3))

def get_dac2_ref():
    rm = visa.ResourceManager()
    scope = rm.open_resource('USB0::0x0699::0x036A::C011565::INSTR')
    dac2_ref = scope.query("MEASUrement:MEAS1:VALue?")
```

```
    dac2_ref_entry.insert(INSERT, round(float(dac2_ref),3))

def get_ck():
    rm = visa.ResourceManager()
    scope = rm.open_resource('USB0::0x0699::0x036A::C011565::INSTR')
    ck = scope.query("MEASUrement:MEAS1:VALue?")
    ck_entry.insert(INSERT, round(float(ck),3))

#TST
Label(win3, text = "V").grid(column = 3, row = 3)
Button(win3, text = "Get Reference Voltage", command = get_tst).grid(column = 1, row = 3)
#Temperaturas
Label(win3, text = "V").grid(column = 3, row = 5)
Label(win3, text = "°C").grid(column = 5, row = 5)
Button(win3, text = "Get Temperature 1", command = get_temp1).grid(column = 1, row = 5)
#ADC_REF
Label(win3, text = "V").grid(column = 3, row = 7)
Button(win3, text = "Get ADC Reference", command = get_adc_ref).grid(column = 1, row = 7)
#DAC_REF
Label(win3, text = "V").grid(column = 3, row = 9)
Button(win3, text = "Get DAC7568 Reference", command = get_dac_ref).grid(column = 1, row = 9)
#DAC2_REF
Label(win3, text = "V").grid(column = 3, row = 11)
Button(win3, text = "Get MAX5725 Reference", command = get_dac2_ref).grid(column = 1, row = 11)
#Clock Frequency
Label(win3, text = "Hz").grid(column = 3, row = 13)
Button(win3, text = "Get Clock Frequency", command = get_ck).grid(column = 1, row = 13)

Label(win3, text = " ").grid(column = 4, row = 2)
Label(win3, text = " ").grid(column = 4, row = 4)
Label(win3, text = " ").grid(column = 4, row = 6)
Label(win3, text = " ").grid(column = 4, row = 8)
Label(win3, text = " ").grid(column = 4, row = 10)
Label(win3, text = " ").grid(column = 4, row = 12)
Label(win3, text = " ").grid(column = 4, row = 14)

Button(win3, text = 'Exit', command = win3.destroy).grid(column = 2, row = 15)
```

8.12 CÓDIGO EM *PROCESSING* PARA ENVIAR E RECEBER DADOS DO ARDUINO

```
import processing.serial.*;
import java.io.*;
int counter=0;
String [] subtext;
Serial myPort;
Float analogValue;
PrintWriter output;
int i;

void setup(){

 //Open the serial port for communication with the Arduino
 //Make sure the COM port is correct
 myPort = new Serial(this, "COM5", 9600);
 //myPort.bufferUntil('\n');
 output = createWriter( "C:/Users/Cátia/SkyDrive/HVRemote Control/data_adc_due_lfsr1.txt" );
}

void draw() {

 readDataFromFile("C:/Users/Cátia/SkyDrive/HVRemoteControl/Dados FPGA/Data_lfsr1_12bit.txt");
 byte out[] = new byte[subtext.length];

 /*Only send new data. This IF statement will allow new data to be sent to
 the arduino. */
 while(counter<subtext.length){
   print("data sent " + float(subtext[counter]));
   print("\n");
   if(float(subtext[counter]) < 0){
     analogValue = (-1)*float(subtext[counter]);
     analogValue = analogValue*51;
     i = 1;
   }
   else{
     analogValue = float(subtext[counter]);
     i = 0;
   }
   analogValue = float(subtext[counter]);
   out[counter] = byte(analogValue);

   myPort.write(out[counter]);
   delay(50);

   //print("Port: " + myPort.available());
   if (myPort.available() > 0 ) {
        String value = myPort.readString();
        Float value2 = float(value);
        if( i == 1 ){
          value2 = value2*(-1);
```

```
        }
        print("data received " + value2);
        print("\n");
        if ( value != null ) {
            output.println(value2);
        }
   }
   //Increment the counter so that the next number is sent to the arduino.
   print(counter);
   print("\n");
   counter++;
 }
 output.flush(); // Writes the remaining data to the file
 output.close(); // Finishes the file
 exit();
}


/* The following function will read from a CSV or TXT file */
void readDataFromFile(String myFileName){

 File file=new File(myFileName);
 BufferedReader br=null;

 try{
   br=new BufferedReader(new FileReader(file));
   String text=null;

   /* keep reading each line until you get to the end of the file */
   while((text=br.readLine())!=null){
   /* Spilt each line up into bits and pieces using a comma as a separator */
   subtext = splitTokens(text,",");

 }
 }catch(FileNotFoundException e){
   e.printStackTrace();
 }catch(IOException e){
   e.printStackTrace();
 }finally{
   try {
     if (br != null){
     br.close();
     }
   } catch (IOException e) {
     e.printStackTrace();
   }
 }
}
```

8.13 CÓDIGO DO ARDUINO PARA ENVIO E RECEÇÃO DE DADOS PARA O MAX189

```
#include <SPI.h>

SPISettings settings(4000000, MSBFIRST,SPI_MODE0);

void setup() {
  pinMode(10, OUTPUT);
  pinMode(12, INPUT);
  pinMode(13, OUTPUT);
  pinMode(9, OUTPUT);
  TCCR1B = TCCR1B & B11111000 | B00000010;
  SPI.begin();
  digitalWrite(13,LOW);
  SPI.setBitOrder(MSBFIRST);
  Serial.begin(9600);
}

byte byte1;
byte byte2;
int bitnum;
float voltage;
float average;
String input;
char inByte[] = " ";
void loop(){
  SPI.beginTransaction(settings);
  while(Serial.available() > 0){
    input = Serial.read();
    //Serial.print(input);
    int count = 0;
    while(count < 351){
      analogWrite(9,input.toFloat());
      //analogWrite(9,102);
      digitalWrite(10,LOW);
      delayMicroseconds(10);
      byte1 = SPI.transfer(0);
      byte2 = SPI.transfer(0);
      digitalWrite(10,HIGH);
      byte1 = (byte1 & 127);
      byte2 = byte2 >> 3;
      bitnum = (word(byte1) << 5) | byte2;
      average += bitnum;
      count = count + 1;
    }
    average = average/350;
    voltage = (float) 5*average/4096;
    //Serial.println(voltage);
    Serial.print(voltage,6);
  }
}
```

8.14 Código do Arduino para Envio e Receção de Dados para o MCP23S17

```
#include <SPI.h>
#include <MCP23S17.h>
MCP expansor(0,10);
void setup() {
  Serial.begin(9600);
  expansor.begin();
  pinMode(8,OUTPUT);
  digitalWrite(8, HIGH); //Fazer enable ao MAX3002
  expansor.pinMode(0B1111111111111111);
  delay(500);
}
int count = 0;
char inByte[] = " ";
void loop() {
 if(Serial.available()){
   char inByte = Serial.read();
   //Escrever para o porto A
    if(count < 8){
      if(inByte == '1'){
        //Serial.println(inByte);
        expansor.digitalWrite(count + 1, HIGH);
      }
      else if(inByte == '0'){
        //Serial.println(inByte);
        expansor.digitalWrite(count + 1, LOW);
      }
    }
    //Escrever para o porto B
    else if(count >= 8 and count < 16){
      if(inByte == '1'){
        expansor.digitalWrite(count + 1, HIGH);
      }
      else if(inByte == '0'){
        expansor.digitalWrite(count + 1, LOW);
      }
      //ler o que foi escrito em ambos os portos
      if(count == 15){
          for(int i = 1; i <=16; i++){
            Serial.print(expansor.digitalRead(i));
          }
          count = 0;
      }
    }
   count = count + 1;
   delay(500);
  }
}
```

8.15 Código em *Matlab* para Geração dos Valores de Teste do Gerador Uniforme Mersenne – Twister

```
clear all;

num_samples = 2^23;
Vref = 5;
N = 12; %number of bits
LSB = Vref/2^N;
%gerador de ruído ideal
U_ideal = num_samples/2^N;
data = RandStream('mt19937ar');
Hist = zeros(1,2^N);

file1 = 'UNIF_MTwister_23.his';
file2 = 'UNIF_MTwister_23.noi';
x = 1;
tic;

while x <= num_samples
  u1= randi(data, 2^N);
  noise(x)= (u1*Vref)/2^N;
  noise2(x) = ((2^N - 1)*noise(x))/Vref;
  bin = int16(noise2(x));
  Hist(bin+1) = Hist(bin+1)+1;
  x = x + 1;
end;

for x = 1:2^N
   if(Hist(x) > 4500)
      Hist(x) = Hist(x)/2;
   end
end

toc
dlmwrite(file1,Hist,',');
dlmwrite(file2,noise,',');

mean_value = mean(Hist);
variance = sqrt(var(Hist));

figure(1);
bar(Hist);
xlabel('codes');
ylabel('samples');
```

8.16 Código em *Matlab* para Geração dos Valores de Teste do Gerador Uniforme Congruente Multiplicativo

```
clear all;
% Number of Samples
NSamples = 2^23;
N = 12;
Vref = 5;
U_ideal = NSamples/2^N;

% Multiplicative congruential generator
myStream=RandStream('mcg16807');

ADC_file3 = 'UNIF_MulCong.his';
ADC_file4 = 'UNIF_MulCong.noi';

Hist2 = zeros(1,2^N);
xy = 1;

tic;

while xy <= NSamples
  u1= randi(myStream, 2^N);
  noise2(xy)= (u1/2^N)-Vref;
  xy = xy + 1;
end;
mi = min(noise2);
ma = max(noise2);
xy = 1;
while xy <= NSamples
  if(noise2(xy) == mi)
     Vin = 0;
  else
     Vin = noise2(xy) - mi;
     Vin = (Vin*5)/(ma-mi);
  end
  Vin = ((2^N - 1)*Vin)/Vref;
  Int_Code = int16(Vin);
  Hist2(Int_Code+1) = Hist2(Int_Code+1)+1;
  xy = xy + 1;
end
figure(2);
bar(Hist2);
xlabel('codes');
ylabel('samples');

dlmwrite(ADC_file3,Hist2,',');
dlmwrite(ADC_file4,noise2,',');

time = toc  % Time in seconds
```